SHARP パソコンテレビ X1
私の勉強ノート
HuBASIC
品川ゆり・Dr.Bee

表紙——品川ゆり
衣裳協力——桂由美ブライダルハウス・鈴屋・ウィークエンズ・コムサデモード
ベンチ・テーブル協力——パックエイジ
スタイリスト——橘　陽子
ヘア・メイク——八田光則
PHOTO——菊地常則
ロゴタイプ・装訂——小林茂二
本文イラストレーション——伊藤アシュラ

SHARP パソコンテレビ X1
私の勉強ノート
HuBASIC
SHARP
品川ゆり・Dr.Bee共著

BASIC言語とは、これまで機械語(0と1の連続で表現)で簡単な計算でも大変な長さのプログラムになったものを、人間の言葉に近い言語で会話しながらプログラムを作ることができる、という素晴らしいコンピュータ言語なのです。

HuBASICは、さらにそのBASIC言語を基礎にして、より使い易く開発されたBASICで、このHuBASICを学習するためには、むずかしい知識は全く必要とせず、ちょうど皆さんが英単語の基礎を勉強するのと何ら変わるところはありません。

その上、単語のスペルや文法の間違いはコンピュータがちゃんと指摘してくれますから、理解・上達が早く、本当に楽しい学習になることでしょう。

さて、パソコンに興味を持った諸君！それではゆりちゃんと一緒に、早速素晴しいX1 Hu-BASICの世界へご案内しましょう。

Dr. Bee.

プロローグ

今，マイコンがブームを呼んでいるようですが，皆さん興味はおありですか？

私は，現在大学で法律の勉強をしているのですが，コンピュータって意外なところでかかわってくるものなんですね。講義の中で，過去の判例をコンピュータに記憶させて，分野別に，あるいは裁判長，検察官別に必要に応じてひき出せるという話を聞いたんですよね。

わぁ，便利って感動したんだけれど，アメリカではもう実用化されているんですって。裁判だってしょせん人間がすることなんだから，裁判官の考え方にも傾向はあるだろうし，規模によっても差がありますよね。

こんな条件でこの程度なら勝てそうだとか，被告の立場は弱いけど過去にこんな弁護をしたら勝訴の判決が出たとか，そういうことがさっとわかれば裁判だって変わってくると思いません？

これから司法を志す人がいるのなら，コンピュータは絶対必要ですよ。そう思って勉強を始めたんだけどなかなか思うようにはいかなくて……数学が嫌いで文系に進んだくらいだからなぁ。

とにかく始めてみようと思って，コンピュータの本を買い集めました。この時，初めて知ったのが，コンピュータに言語があるという事。え，言語？　と思うでしょう。

どういうことなのかって言いますと，つまり，世界中には英語を話す人もいれば，日本語，独語，仏語を話す人もいるし様々ですよね。どの言葉でもお互いに同じ言葉を使えば，意志は通じるけれど，一人が英語，一人が日本語ではだめですね。コンピュータもこれと同じだという事なんですよ。

言語の種類には BASIC，コボル，フォートランなどがあります。それぞれの言語にそれぞれの文法があると考

えてもらうと理解しやすいんじゃないかしら。文法を間違えたら意味はわからないし，ましてや，他のことばの文法と混ざってしまったら大変なことになりますよね。

そんなわけで，まず言語を決めて，正しい手順で勉強することが必要となってきます。

今度私がトライしてみるのは BASIC。B=Beginner's A=All-purpose S=Symbolic I=Instruction C=Code というわけで，つまり，“初心者向きであらゆる目的に合う記号を使った命令語” なのだそうです。早い話が初心者向きということで，一番入りやすいのではないかと思ったのですが……そう簡単にはいかないみたいだなぁ。

さらに，BASIC といってもいろいろあるようなのですが，私が勉強するのは HuBASIC です。機械がシャープが新発売したパソコンテレビ X-1 なので，これを選びました。

Hu-BASIC の Hu は HUDSON SOFT の HU で Humanity の Hu でもあるそうです。ちょっと素敵でしょ。別にコンピュータは，非人間的なものでも冷たいものでもないんですよ。

“新しく何かを始めるには，まず実際に「やってみる」ことです。でも「やってみる」といっても，初めての人にとってはなかなか勇気のいることだし，第一やっぱりしんどいものです”。

“水泳の本を読み，畳の上で練習したからといって，すぐに海に飛びこんだら溺れてしまいます。それは知識をたくわえても，「勘」がついていないからです。「勘」をつけるには，やっぱり基礎から勉強してスマートに「やってみる」ことです”

“コンピュータは少しぐらい使い方をまちがっても，あなたを危険な目にあわせるようなひどい道具ではありません。使い慣れたら手離せないくらい便利な道具です”。

この文は同じシャープの MZ-1200 のマニュアルから引用したものですが，最初どうしようかと迷っていた私にヤル気を起こさせてくれたとっても印象深い言葉です。

そういうワケで，結局まずは手で機械に触れ，くり返しやってみなくちゃならないというので，考え抜いた揚句，とにかく HuBASIC を開発した HUDSON SOFT の東京営業所を訪ねてみることにしたんです。

その時指導して頂いたのが，実はこの本の共著である Bee 先生なんですね。Bee 先生の優しく根気強い指導のおかげで，今や私も独力でプログラミングが組める程に成長しました。ホントですよ!!

で，この私の成長過程を綴ったのが「パソコンテレビ X 1 HuBASIC 私の勉強ノート」ってワケなんですが，これは私が Bee 先生のおっしゃることを大学ノートに綴ってたのを Bee 先生が見つけ，これは面白いといって一冊の本にしようと言い出したのがキッカケ。

基礎になってるのが私の大学ノートにビッシリ書き込まれたメモですから，何となく心もとないのですが，足りない分は，欄外に Bee 先生のコメントを入れフォローするというので OK しました。また欄外の空白は読者の皆さんがメモを書き込む様にも配慮されたスペースだそうです。

こうして，あれこれ戸惑いながら（何日か徹夜もしたんですよ）完成したのがこの「パソコンテレビ X 1 HuBASIC 私の勉強ノート」です。

少しでも読者の皆さんが，パソコンテレビ X 1 を使う上での，また BASIC を理解する上での手助けになれば幸いです。

前置きが大変長くなってしまいましたね。そろそろノート 1 「まず，X 1 の構造を知りましょう」から入って行きましょうか!!

目次

NOTE1 まず、X1の機能を知りましょう!!

次代のパソコンに新機軸、その超合理的な開発意図は……

①パソコンディスプレイとTV画面の共用……………………8
②8ビットの頂点に立つＸ１……………………………12

NOTE2 さあ電源を入れてみましょう

まず電源をONにして、HuBASICをロードしないとプログラミングはできません

①HuBASICのローディング ……………………20
②パソコンによる計算の手順 ……………………24
③変数を使いこなすテクニック……………………29

NOTE3 いよいよHuBASICに入ります!!

HuBASICの基本構造を学んで、BASIC言語の全体像を掴もう

①プログラムモードの勉強 ……………………32
②コンピュータ言語と論理構成 ……………………39
③ゲーム・パズルの征服……………………47
④FRACTION処理の利用 ……………………54
⑤FOR～NEXT文とステップ命令の扱い方……61
⑥カズアテゲームの構造 ……………………72
⑦PRINT文、IF文、DIM文の応用 ……………75

NOTE4 文字列を中心とした特殊関数

文字列(STRING$)は、関数の基本。これを使いこなしてプログラミング力をアップ

①STRING$の基礎と応用 …………………………… 84
②LEFT$の特殊性と利用法 ……………………………86
③STRING$とFOR～NEXTの組み合わせ…………91

NOTE5 関数の征服にチャレンジ

いろいろな関数を、この際徹底的にものにしてしまおう

①ASCIIコードの原理と応用 ……………………………97
②CHR$とASC関数の関わり ……………………………100
③GOSUB文とRETURN文 ……………………………105
④数値定数と指数部、仮数部………………………… 110
⑤VAL関数とSTR$の関わり …………………………112
⑥時計命令の作り方と実行 ……………………………115

NOTE6 関数の整理と特殊コマンドの勉強

関数と特殊コマンドの扱い方を覚えると、プログラミングの世界が大きく拡がる

①数値関数の種類とまとめ…………………………… 122
②フローチャートの考え方 ……………………………128
③周辺機器の制御に貢献する命令 ……………………132
④プログラムのループを制御する命令 ………………142
⑤プログラムの整理・統合を助ける命令………………145

NOTE7 コンピュータサウンド

パソコンでテクノサウンドを演奏してみる

①プログラミングのための音楽記号 …………………152
②TEMPO命令、SOUND命令、和音構成の方法… 155

NOTE8 グラフィック処理に挑戦!!

HuBASICを学ぶ上で最も楽しいコンピュータ・グラフィックス

①グラフィックの中心PALET …………………… 160
②SCREENと画面構成 …………………………… 163
③PSETとPRESET…………………………………168
④LINE, BOX, CIRCLEのグラフ命令 ……………172
⑤HEXCHR＄とWINDOW命令……………………178
⑥プリンタに関する命令……………………………186

NOTE9 サンプル・プログラム

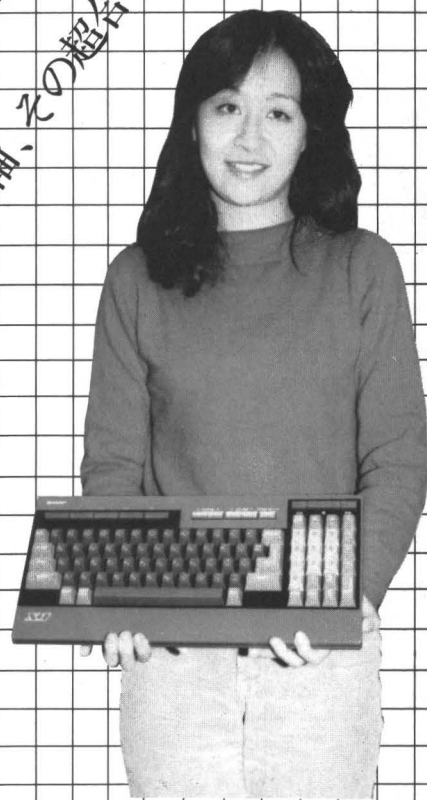
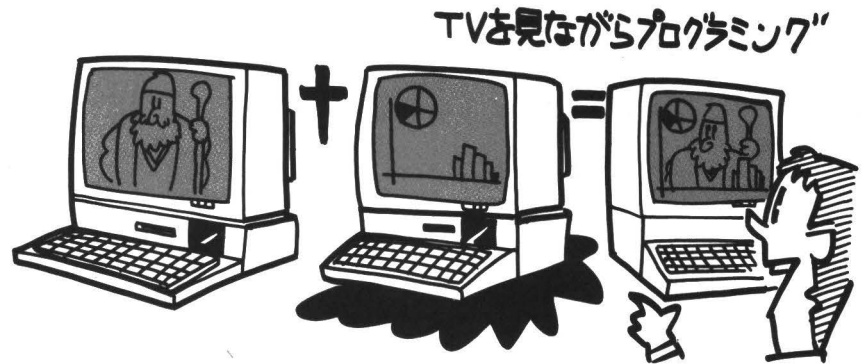
学習システム、図形処理、実用メモ、ミュージックの応用プログラム集

①日本国憲法………………………………………… 197
②世界の国々………………………………………… 216
③テレフォン・リスト………………………………226
④ミュージックプログラム…………………………231

NOTE10 X1 HuBASICの概要

その最高のハードウエアと最高のソフトウエア

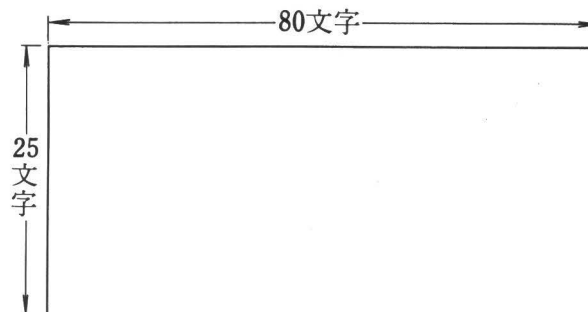
①X1 HuBASICの特徴 ………………………………238
②一般コマンド・ステートメント …………………241
③その他の命令、関数、予約変数 …………………244
④エラー・メッセージ………………………………250
⑤その他(コード＆マップ) …………………………252

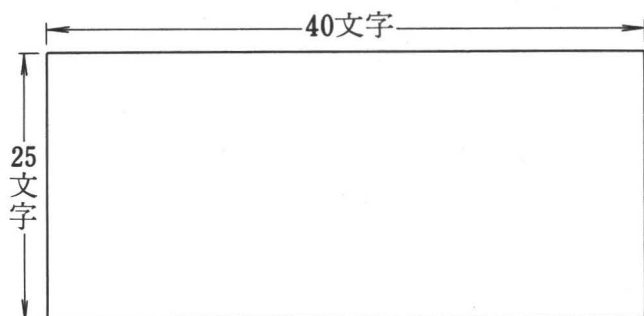
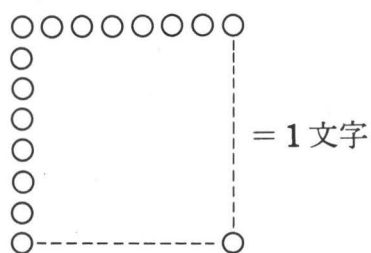
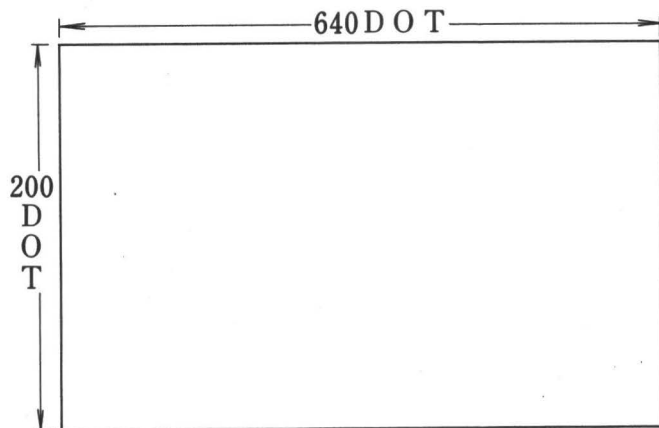
NOTE1
まず、X1の機能を知りましょう!!
次代のパソコンに新機軸、その超合理的な開発意図は…

コンピュータのディスプレイ画面をTV画面としても使えるの。いろいろ秘訣も…

さて，私の使うコンピュータは……。　え，これがコンピュータなの？　見た感じはまるでテレビよねえこれじゃ。

このＸ１は，TV も視られるコンピュータなんですって。TV コンピュータとでも言うべきかナ。要するにディスプレイをコンピュータの CRT 画面としても，TV のブラウン管としても使えるの。だからパソコンとテレビが一体化したものなのね。

キャラクタはDOTという画素単位で構成されていて、2000文字も出ます

CRT ディスプレイの説明をしておきましょう。画面はどうやって構成されているのかしら？

このように，画面の横一杯に文字を打つと80文字，縦は25文字入れることができます。これを80モード（横に80文字入るモード）と呼びます。

80モードに対して，40モードというのも，あるそうなんですが，これは1文字が80モードの文字の倍の大きさだと考えれば良いわけです。同じディスプレイに，80モードの半分の文字しか，入れられません。

さて，画面を更に細かい単位で見てみるとDOTという単位が出てきます。1DOTは画面を構成している最小単位です。

こんなふうに，1文字は，縦，横8個ずつ計64個のDOTで区切られています。だから，ディスプレイ全体を，DOT数で表わすと，こうなるでしょ。

できるだけ多くのDOT数を表示できることがパソコンディスプレイの必要条件です

そこで，図のように最低640×200 DOTというDOT数の表示を可能にするためには，どのような問題を解決しなければならないかということです。

ここで，とくに問題になるのが，水平方向のDOT数，つまり640 DOTということ。

なぜ，この640 DOTが問題にされなければならないかというと，一般家庭で使われているテレビ受像機では，こんなにたくさんのDOT数を表示できないからだというんですネ。

ここから先の話は，法学部の私にとってはチョット荷が重いのだけど，いろいろと受け売りの話も混ぜて，この問題をさらに煮詰めていきましょう。

家庭用のテレビでは，放送されてくるテレビの映像周波数帯域の上限が4.2 MHz止まりだってこと，知ってる？

このことから，ディスプレイ画面，いやブラウン管上に表示できる画像の解像度は，せいぜい250本から300本どまりになるんだそうです。この300本の解像度というのは，画面上に細い縦の直線を何本も表示させてみて，いったい何本の直線までを見分けられるか，というところで，その画像表示能力の細かさが決定されるというわけ。だから，その数は多い方がいいというわけネ，当然のことだけど。

しかもですヨ，これは，特別に性能の良い受像機を使ったらの話しなんだから。平均的なテレビの性能というのは，“推して知るべし”なのよネ。

これで，家庭用テレビでは，水平方向640 DOTを表示するのはとても無理だということがよく理解してもらえたと思うけど。

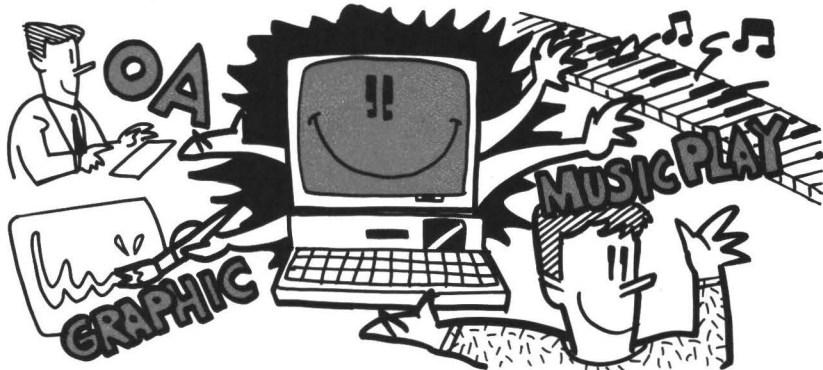
それでは垂直方向の200 DOTはどうかといいますと，日本

のテレビ方式では，画像のもとになっている水平走査線が525本，視える部分でも500本近い数を確保しているから，200 DOTを表示するのはワケないのよネ。だから，問題になるのは，あくまで水平方向の640 DOTというわけなの。

そこでなんだけど，この640 DOTを表示しきれるのは，専用のパソコン用カラーCRTディスプレイしかないということになるわけネ。

たしかに，このX１の内容と実力からいえば，そこいらのテレビで表示しきれるものではないとは思うのですヨ。でも，それではパソコンテレビにならないわけです。

ここで，もう一度確認しますけど，このX１というのは，パソコン部CZ-800Cとディスプレイテレビ CZ-800D が組み合わされたものです。

だから，このテレビ部分CZ-800Dは,ディスプレイテレビと呼んでいるけど，画像表示能力のレベルでは，専用ディスプレイ並みである必要があるってことなのネ。

RGB入力端子とビデオ入力端子を付けたディスプレイで万能の映像機器に

さあ，いよいよこのディスプレイテレビの本題に入るんだけど，構造と機能からみていくと，テレビ受像機とパソコンディスプレイを兼用させるために，いろいろな工夫が盛り込まれていることがわかります。

その第一は，RGB カラー三原色分離独立入力端子が付いていること。

もう一つ余談になりますけど，入力端子としては，複合映像信号入力，通常はビデオ入力と呼ばれる端子も付いているのです。

ここで，回り道をしておさらいといきましょう。テレビ受像機の構造をみると，アンテナから入った電波の通る回路を順番に並べると，高周波（RF）回路→映像中間周波増幅（IF）回路→映像増幅→カラー復調（分離）→CRT という道筋になるのだそうです。

映像別に入力端子を別々にするというのがX1の基本コンセプト

で，ビデオ入力というのは，映像中間周波と映像増幅の間に，またRGB 入力というのは CRT 直前に入力するようになっているそうなの。

そこで，話しを単純化すると，CRT に近いところから入力する映像ほど，純度が高くて，質の良い画像が得られると言えるのじゃないかしら。

だから，RGB 入力は，パソコンディスプレイにとって，もう絶対に必要だということなのよね。

それから，テレビ放送電波なんだけど，これはアンテナから入って来るのよネ。当然だけど，X 1にとっては第三の映像というわけ。ビデオ入力はホームビデオやビデオディスクなんかの専用入力口。パソコン用入力は RGB だから，それぞれ情報別に，別々の入口から情報を取り入れるというのがX 1の基本コンセプトなのよね。

こういう思想は，最近のモニターテレビなんかにもみられる傾向なの（なあんてエラそうにいったりして)。X 1のような，あるいは CZ-800 D のような，三系統の入力端子を持ったディスプレイテレビというのは，一種の流行になりつつあるといえるのだけど。

RGB入力はCRTにストレートに情報を入れることが可能、合理的なのです！

ま，それはともかく，RGB 入力の強みは CRT に近いところから入力できるということも大きいけど，そのため，従来のテレビ用回路の限界に制約されることなく，最高 15 MHz の高周波帯域で映像情報を送り込めるというのが大きなメリットなのよネ。テレビのように 4.2 MHz がせいぜいなんていうのとはもう天と地ほどの違い。

だからこそ，80×25字の 2000 字もの文字表示もできるの，なんて，もうここでしつこく繰り返すまでもないわよネ。

シャドウマスクの目を細かくしたファインピッチ管を採用、TV用にも配慮!

しつこいついでに，シャドウマスクの話しもひとつ。シャドウマスクのことは，ここではくわしくは説明する必要はないと思うけど，カラーテレビにはなくてはならないものということだけは，確認しておきましょう。

このシャドウマスクは，網目状になっているんだけど，この網目がある程度細かくないと，解像度の高い画像を得られないのです。

従来のテレビでは，0.63 μm ピッチという目の細かさが基準だったらしいの。これは，テレビでは，それほど細かな画像を必要としなかったことと，逆にあまりピッチを細かくすると，ブラウン管の明るさが十分に得られなくなるという欠点が出てくるためでもあったんだって。

ところが，専用カラーディスプレイでは，明るさよりも，解像度優先なんだから，シャドウマスクは，ギリギリまで細かくつめた高精細度管を使っているわけ。

そこで，CZ-800 D では，標準のシャドウマスクピッチを更に細かくしたファインピッチ管を採用して，解像度のレベルを，できるだけ専用ディスプレイに近づけ，しかも，CRT 表面のフィルタ塗料も新しいと言うことですから，見かけ上の明かるさコントラストは従来のブラウン管以上。だから，このＸ１でテレビを観ても従来のテレビに勝るとも劣らないというか，とにかくきれいな“絵”が楽しめることうけあい……なんだって。

SHARP
COMPUTER/TV
COMPUTER
∨ CHANNEL ∧
∨ VOLUME ∧
POWER
CZ-800DW
CENTRAL PROCESSING UNIT
POWER
SHIFT
CTRL
BREAK
SHIFT

X1は8ビットの頂点に立つ、究極的な8ビットマシンなのね!!

ホ！やっと一段落ネ。ひととおりの説明を無事に終えてやれやれといったところ。それにしても，なれない説明で肩が凝ってしまったワ！

でも，いっしょうけんめい説明したかいあって（なぁんて自分で言ったりして），このX1が，いかに新しい情報機器としてふさわしい内容を備えているかということが，よく理解していただけたと思うの。

あんまりテレビ部分ばかり力を入れて説明してしまったので，肝心のパソコン部分，つまりCZ-800Cについての説明が不足したようネ。

ただ，テレビ部分のような出力機器自体が高性能でしっかりしていないと，X1のように，グラフィック分野に力を入れているパソコンの能力を十分に発揮できない，ということをおさえておきたかったの。そこのところを皆さん了解して下さいネ。

さて，CZ-800Cの話しに戻りましょう。CZ-800Cは「8ビットの頂点に立つ」究極的8ビットマシンという開発コンセプトで始められただけあって，今までの8ビットマシンの集大成といった優れた内容と使い易さを備えているのです。

何が「8ビットの頂点」かと言うと，このX1は，8ビットマシンが一度にアクセスできるメモリ容量64Kバイトという限界をつき破って，128Kバイトまでアクセスできる能力を持たせたということなんです。しかもバンク切換えなしで，純なかたちに使える8ビットマシンなのです。

これは，むずかしく言うと，ソフトウエアを工夫することによって，I/O空間をメモリ空間に開放することで，従来の2倍の128Kバイトというメモリ数をアクセスできるようにしたということで，これによって，ハイスピード化，機能アップなど，「これで8ビット?!」というくらいの充実した内容を誇るのだそうです。

SHARP
F 1
F 2
F 3
F 4
F 5
CHANNEL
VOLUME
COMPUTER/TV
ESC
BREAK
INS
DEL
CLR
HOME
SHIFT
SHIFT
CAPS
LOCK
カナ
7
8
9
4
5
6
1
2
3
0

シャープ自信のフルRAM構成、ソフトウェアに期待がかかってます

さて，この128Kバイトのメモリですが，4KバイトのIPLを除いて残りは全てRAMつまりフルRAM構成になっていることです。

フルRAM構成では大胆に推し進めてきたシャープも，よくもここまで思い切って…というくらいの企画だと思うのです。

メモリの殆んどがRAM，これをみただけでも，今後のソフトウエア開発面での自信，それにユーザーとしても（もちろん私のように，もともと機械には強くないヒトでも），一度使い始めると，もうグッと引き込まれて自分でどんどんコンピュータのとりこになっていってしまいそうな柔軟さと深みがあるんですよネ。

ですから，世の中では16ビットマシンに期待がかけられているという傾向もみえますが，私は，一度でもこのX1に触れると，もう16ビットだ何だのと言う前に，どうしても8ビットを完全に征服してみたいという気持ちに駆られるのも当然のことだと思うのです。

テレビとパソコンが同時にディスプレイ上に映し出せるスーパーインポーズが特徴

　ということで，このＸ１の実力について，私なりに目につくポイントを挙げてみました。ここで，もうひとつ付け加えたいのは，スーパーインポーズという独特な機能です。

　最初のところで，Ｘ１では，テレビとパソコンディスプレイの機能が矛盾なく同居することができるよう，さまざまな面で工夫が凝らされていることを説明しましたネ。

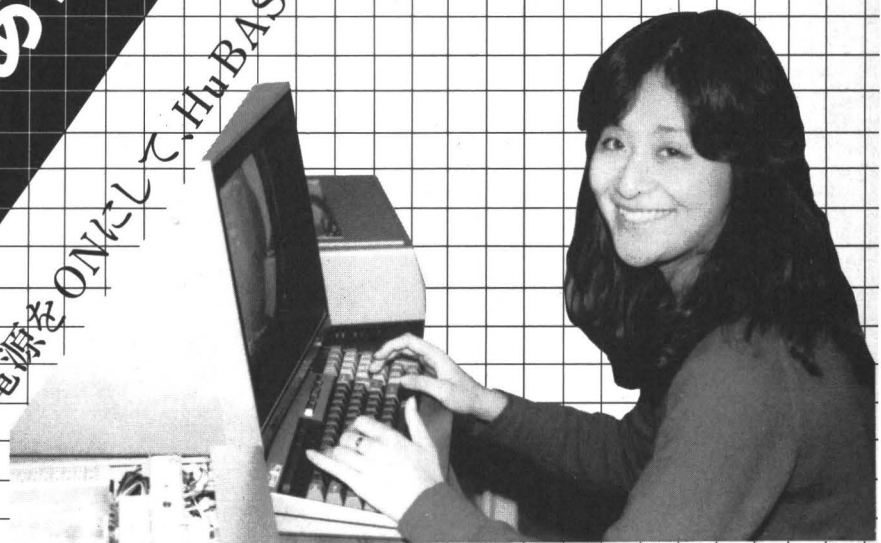
　しかし，それはテレビとパソコンディスプレイが，それぞれ別々に十分な機能を果たすということを第一としていたのです。

　このスーパーインポーズは，そこから一歩進んで，テレビとパソコンを同時に一つのディスプレイ画面に載せるという考えです。両方の画面を合成できるようになると，またいろいろな楽しみが出てくると思うのですよネ。

　技術的な問題点は，それほど障害にはならないと思うのですが，例えば，テレビやビデオ画面にはインターレース方式という画像の構成のし方，パソコンでは，そのインターレースがない，というように，方式の違いがあるそうなんです。これについては，細かな説明はしませんが（誰ですか！できないんだろうなんていう人は），こういった方式間の違いも，最終的にはうまくまとめてしまったということですから，このＸ１，商品企画をコンセプトのままで終らせることなく，絶対に実用的な商品にするんだ！という製作者側の情熱がさまざまな障害を乗りきったといえましょう。

　私もこれから実際のＸ１の使い方についてもっと深く探求していくことになります。

　その過程で，一層Ｘ１のすばらしさに出会うことができると思うのです。

NOTE2
さあ電源を入れてみましょう
まず電源をONにして、HuBASICをロードしないとプログラミングはできません

ＣＲＴディスプレイと本体の電源をＯＮ！　あれ，何か出てきましたよ。

Make ready any device
Push (F, R, C or T) Key
F : Floppy
R : ROM
C : CMT (CASSET MAGNETIC TAPE)
T : Timer

こんな表示が現われます。さて，Ｘ１を SHARP HuBASIC の使える機械にするには，どうしたらいいんでしょうか。

まず、BASICテープをロードするところから。Cキーを押してテープスタート

まず BASIC テープを入れ，上の表示に合わせて，どれかの KEY を押さなければなりませんね。カセットを利用するわけだからCキーを押します。

この時，カナ KEY がロックされてないかどうか確かめるのを忘れずにね。OFF の状態になってないとCキーは押せません。ソを押してることになっちゃいますよ。

IPL is looking for a program from CMT

まずこんな表示が現れ，次に

IPL is Loading BASIC CZ8CB01

という表示に変わります。こうなるとロード中なわけで，１

〜2分程ロードが終わるのを待ちます。X1だと，すぐテレビと切り換えられるので，こんな時はテレビでも見ながら，のんびり待ってください。

```
SHARP-HuBASIC CZ-8CB01 V1.0
Copyright (C) 1982 by SHARP/Hudson
```

```
 23536 Bytes free
Ok
■
```

ロードが完了すると、BASICが使えます。キー入力の後、CRキーを忘れずに!

こうなったらモードを切り換え（テレビをコンピュータに切り換え）カセットを取り出してください。

はい，もう準備完了！

これでX1は BASIC が使える機械になったわけです。さて，何か書いてみなさいという先生のお言葉により，早速自分の名前を打って見ました。（☞1）

```
Ok
shinagawa yuri ■
```

BASIC では，入力が終わると，必ず CR ボタンを押さなければならないということなので，

```
Ok
shinagawa yuri CR
Syntax error
Ok
■
```

1
コマンド(命令)を受け付けるのは、
▒
画面のカーソルが点滅している時にのみ受け付けるのです。

BASICの文法に合わない入力をすると、Syntax errorの表示が出るのです。

何でしょうねぇ…。この Syntax error って？　エラーは誤まりだっていうのは，想像がつくんだけどね。えっーと……Syntax とは？　辞書によりますと，構文法，文法的語句配列などとあります。

なるほど。つまり私は，何か文法的な間違えをしちゃったわけか。でも BASIC の文法って何なのかしら。

先生に言わせると，表示の仕方一つにしても，きちんと方法があるんだそうです。

まあ，まだ何も教わってないんだし，仕方がないですよね。

PRINT命令は、入力文を画面にあらためて書き出しなさいという命令なのです

PRINT 命令とは，画面に何かを書きなさいという命令なんだそうです。

さっき間違えた私の名前を正しい文章で書くと，

```
print "shinagawa yuri" CR
shinagawa yuri
Ok
■
```

キャリッジ・リターンのキーは、実際にはこんな型をしていますけど、CRで表わしましょうね。

こうなるのです。

この" "は何でしょうねぇ……。

SHIFT キーと 2"フ キーを同時に押して ” を表示！ 中に入る文字が出力されます

ダブルクォーテーションを表示するには，SHIFT のキーと一緒に 2"フ のキーをおすのだそうです。ダブルクォーテーションとダブルクォーテーションの間にはさまれた文字を，コンピュータは出力してくれるようですヨ。

ダブルクォーテーション

” ＝ SHIFT ＋ 2"フ

パソコンを電卓代わりにして計算しましょう!演算用記号は＋－＊／など

では，今度は電卓代わりに使ってみましょう。じゃあ，PRINT 文で電卓がわりに計算させてみます。コンピュータ君できるかしら？（☞2）

```
print "5+3" CR
5+3
Ok
■
```

あれ、変だなあ

```
print 5+3 CR
 8
Ok
■
```

そうか!! ダブルクォーテーションは文字を書くときに使うもので，計算をする時はいらないのネ！

演算子	＋	たし算
	－	ひき算
	＊	かけ算
	／	わり算

コンピュータでは，たし算，ひき算は普通と同じだけれど，かけ算(＊)わり算(／)は，こんな記号を使うんですねぇ。

2
PRINT命令は，省略形が使えますよ。(P.)(？)を書いても同様に判別してくれます。

```
?5+3          P.5+3
 8             8
Ok            Ok
```

たし算、ひき算もBASICの手順で式を……、四則演算の順は算数と同じよ

ところで，先生から，誰でもわかるような質問を受けました。
「2たす3はいくつですか？」

私は「5に決まっているじゃないですか」と答えました。
「では，それに2をかけたらどうなりますか」と先生。

当然「10」と答えますよねぇ。「それじゃ BASIC で書いてみてください。」

```
print 2+3*2 CR
 8
Ok
■
```

あら，どうしてかな？ <こんな計算でもうわからなくなっちゃうのかしら？>あっ…！ もしも，算数と同じように四則演算に順番があるとしたら，「かっこ」がいるんじゃないかしら。ちょっとためしてみましょう。

```
print (2+3)*2 CR
 10
Ok
■
```

やっぱり。我ながら，そそっかしいなぁ…。

まったく算数と同じで，かけ算，わり算は先に実行し，たし算，ひき算は後になるわけなんですね。なんだ算数も BASIC も同じじゃない！

YURI の感想……

何をやるにもそうだと思うけど，ある程度のルールという物は必要ですよね。ルールの1つ1つにひっかかっていては，ルールを超えた本当のおもしろい部分に触れることはできません。

いえ，きっとそうだと思うんですよね。元になるルールを早く覚えて，はやくおもしろいプログラムを作ってみたいですよね。一緒にがんばりましょう。

電卓のメモリと同じような使い方。A=1はAというメモリを1にすること

```
a=1
Ok
b=2
Ok
c=a+b
Ok
print c  CR
 3
Ok
■
```

（☞3）

電卓のメモリと同じような使い方があるって聞いたけど，どういうことなんでしょうね？

A＝1はAというメモリを1にしたいという事でBも同じようにわかるんだけど，C＝A＋Bってどういう事かなぁ。なぜA＋B＝Cじゃいけないのかしら。

BASIC では求める値つまりCを先に書くのが決まりなんですって。Cは結果じゃなくて，求める値だという事みたい。

でも，私ならいきなり PRINT A＋B って書いちゃうだろうなぁ。

```
a=1
Ok
b=2
Ok
print a+b
 3
Ok
■
```

これでも，答えは出ますよねぇ…。

3

変数への代入、計算式等は、正確にはLETと付けるのですけれど、HuBASICでは、そのすべてを省略することができます。

```
LET A=1
Ok
LET B=2
Ok
LET C=A+B
Ok
PRINT C
 3
Ok
■
```

求める値Cを先に書いてC＝A＋B。こうするとC−A，C−Bとやれて検算が簡単

勝手な事をしちゃいけないと先生にしかられてしまいました。

先生は検算するつもりだったんですって。検算するってどんな事なんでしょうねぇ…。

```
a=1
Ok
b=2
Ok
c=a+b
Ok
print c
 3
Ok
print c-b
 1
Ok
print c-a
 2
Ok
■
```

なる程ね。それでCという値にA＋Bを入れて置いたのか。ただ PRINT A＋B でも答えは出てくるけど，こんな風に検算はできませんよねぇ。……納得！（☞4）

HuBASIC では私の名前も**変数**になってしまうんだって!!

```
shinagawa=15
Ok
yuri=5
Ok
watashi=shinagawa+yuri
Ok
print watashi
 20
Ok
■
```

こんな使い方もできるのねぇ。

4

変数は、240文字までの英数文字の組合わせを使うことができます。

```
SHINAGAWAYURIWAKEIOD
AIGAKU=1
Ok
▒
```

なんてことも出来ますが、間にスペースを入れることは出来ません。

又、頭に来る命令文を変数にすることもできません。

```
PRINT=1
Syntax Error
Ok
APRINT=1
Ok
▒
```

A=A+1とは、Aの再定義をすること、コンピュータは、再定義後の値を記憶

```
a=1
Ok
a=a+1
Ok
print a
 2
Ok
■
```

変な式だなぁ？　どういう事？　はじめにA＝1だったのにどうしてPRINT Aは2になるのかなぁ。ここでは，最初に出て来たAとA＝A＋1のAとでは別の条件になってしまったと考えれば良いみたいです。

コンピュータは再定義された新しい方のAの値を覚えているわけなんですね。

文字も変数として使えるの、変数の最後には必ず$をつけましょうね

文字も変数として使えるっていう事なんだけど，今度は一体どんなことを教わるのかしら？

どう考えても，文字は文字だし，数字は数字よね？　**文字変数**なんて聞くと，なんだか古典と数字が一緒になったみたいな変な感じがするなぁ…。

大体数字はとっつきにくいし，だからコンピュータもちょっとね…と思ってたんだけど，考え方を変えるべきかしらね。文字を変数にする時は必ず，変数の最後に$をつけるらしいんですが，ちょっとやってみましょうね。(☞5)

```
a$=shinagawa
Type mismatch
Ok
■
```

あれ，間違えちゃった。何がいけなかったのかしら…。そうだ！　文字を書く時はダブルクォーテーションを使うんだって確か前に教わったのよね。それではやり直し。

```
a$="shinagawa"
Ok
b$="yuri"
Ok
c$=a$+b$
Ok
print c$
shinagawayuri
Ok
■
```

5
Type mismatch
変数の型が一致していない時に出るメッセージです。このエラーが出たら文字変数か数字変数かよくたしかめてくださいね。

C$=A$+B$、こうすると、文字も変数になってC$はshinagawayuriに

できた!! でも文字が変数になるっていう意味がよくわからないなぁ…? 数字の変数と同様に再定義なんていう事もできるのかしら?

```
a$="shinagawa"
Ok
a$=a$+"yuri"
Ok
print a$
shinagawayuri
Ok
■
```

なる程，こういう事が可能だというわけね。つまり，ほとんどの考え方は，数字の変数と同じで，ダブルクォーテーションを忘れないでつけるという事ね。

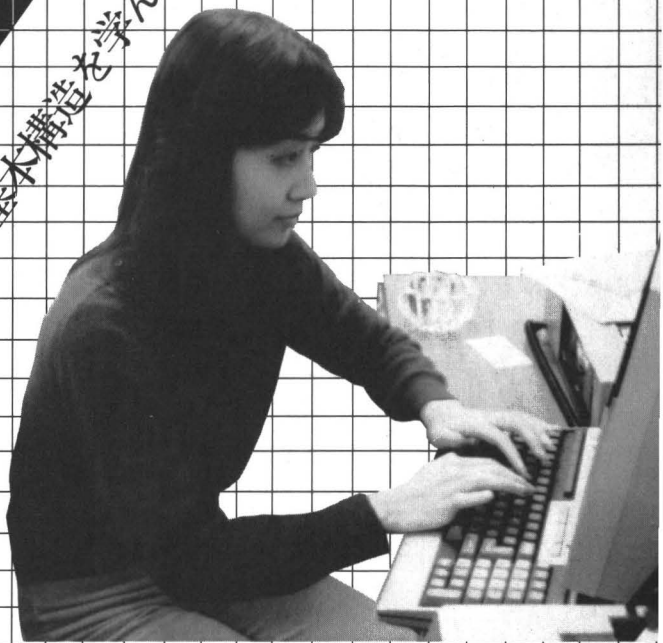
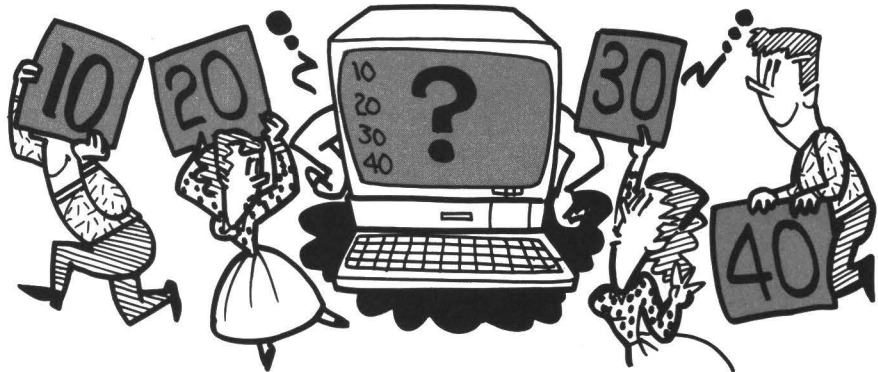
```
a$="shinagawa"
Ok
b$="yuri"
Ok
print a$;b$
shinagawayuri
Ok
■
```

；このマークは，セミコロンって言うんですって。直接PRINT で書かせる時には；で続けてもいいみたい。(☞6)

ここで，頭文字のところだけ SHIFT キーを押しながら打ってみましょうね。それに Yuri の前にスペースをあけて打った方が見やすいでしょ。文字変数の場合は，A＄＋B＄は文字をつなぎあわせるだけの事だから，セミコロンと同じ事です。

```
a$="Shinagawa"
Ok
b$=" Yuri"
Ok
print a$+b$
Shinagawa Yuri
Ok
■
```

SHIFT キーを使って、
頭文字だけ大文字にします

NOTE3
いよいよHuBASICに入ります!!
HuBASICの基本構造を学んで、BASIC言語の全体像を掴もう

プログラムモードの勉強から入ります。ダイレクトモードとは区別してね

さあ，いよいよプログラム・モードの勉強に入ります。これがわかれば，自分でプログラムが作れるようになるわけよね。がんばらなくっちゃ!!

まずプログラムモードとダイレクトモードは別のものだっていう事を頭に入れておいてください。それじゃ実際にプログラムを作るのに何を知っておかなきゃいけないのかナ？

まず大切なのは，行番号といわれる実行の順番を示す番号と命令語が必要だという事らしいんですよね。行番号っていうのは10番から10番おきに書いていくのが一般的なんですって。

ダイレクトモードとの違いは、行番号と命令語が必要なこと。そこでRUN!

```
10 print "Shinagawa Yuri"
```

あれ？ 正しく打ったはずなのに出てきませんよ？ ここがダイレクトとの差！ 必ず命令語を入れてやらないと，コンピュータは実行してくれないんですよ，じゃあ，なんて命令すればいいの？ ここで登場するのが RUN !! (☞7)

7
プログラムを**走らす**、プログラムが**走る**、とよく言われますね。
そう、そのRUNなのです。

```
10 print "Shinagawa Yuri"
run
Shinagawa Yuri
Ok
■
```

こんな風になるわけです。もしも何か間違っていてもＲＵＮしないと ERROR は出てこないんです。

```
10 prnit "Shinagawa Yuri"
run
Syntax error in 10
Ok
■
```

こんな風にＲＵＮした後で表示されてくるのです。それに，必ず行番号の順に実行していってくれます。

```
10 print "Shinagawa"
20 print "Yuri"
run
Shinagawa
Yuri
Ok
10 print "Yuri"
20 print "Shinagawa"
run
Yuri
Shinagawa
Ok
■
```

プログラムモードでは、AUTOで、自動的に行番号が入れられるの。さあINPUT！

今は10，20と行番号は自分で入れましたけど，ここで **AUTO** と命令すれば，自動的に10，20，30と行番号が入りますョ。

それでは，今度は実際に，たし算のプログラムを作ってみましょう。＜今までのところ，忘れてないかなあ…もう一度確認してみてください。＞（☞８）

8
AUTOは、行番号を自動的に発生させる命令です。
このAUTO命令を中止するには、SHIFT＋BREAK キーを同時に押すと中止されます。

```
a=1
Ok
b=2
Ok
c=a+b
Ok
print c
 3
Ok
■
```

これは今までよく出て来たダイレクト・モードですよね。

INPUTでは、入力する値を決めなければならないのね。これでC=A+Bの答が…

```
10 input a
20 input b
30 c=a+b
40 print c
```

今度は，プログラム・モードでやってみました。INPUT という新しい命令が出てきたけれど，これは，“情報を与えて下さい”という意味らしいんです。これを実行するには，またRUNを使うのかナァ？　まあやってみましょう！

```
run
? ■
```

あら！　？は何だろう。　どうしたらいいの。＜私って新しいものがでてくると，すぐあわてちゃうのよネ！　まずは落ちついて推理力を働かせる事ネ！＞

？という事は何か聞いているんじゃないかしら。10に打ったのは INPUT A だから，たぶん最初のAの値を決めてくれという事なんじゃないかしら。

じゃあ最初の？を1としてみましょうか。そして入力し終ったら必ず CR を押すんでしたね。

```
run
? 1
? ■
```

あれ？　また？が出てきたなあ，じゃあこれはきっとBの値を聞いているのね。

```
run
? 1
? 2
 3
Ok
■
```

これでやっとたし算のプログラムができたんだわ。これと同じようにかけ算やわり算もきっとできると思います。さっきのプログラムの30，C＝A＋Bの＋の部分を＊，／とかえていけばいいわけよね。じゃあ，わり算にしてためしてみましょう。

もちろん、四捨五入もできるの。コンピュータも0で割るのは不可能

```
30 c=a/b
run
? 2
? 3
 .66666667
Ok
run
? 8
? 9
 .88888889
Ok
■
```

小数点以下は，8けたまでしか表示されないので，9けためが四捨五入されて，上のようになってくるわけなんです，なる程。でもHuBASICというのは16けたまで表示できるって本に書いてあったけどおかしいな？

```
run
? 1
? 0
Division by zero in 30
Ok
■
```

また変なものが出てきちゃった。これは何でしょうねぇ。先生のお話で，やっと納得！　つまり算数で考えてもこんな計算変なんですよね。0で割るなんてことは不可能なんです。コンピュータはちゃんとそれがわかってるのでエラーとして表示してくれたわけね。<フーム！　すばらしい>（☞9）

9
ここにどの様な時でも0で割ることはゆるされません。あとで勉強しますが，もし変数が0になるような時にはその計算を行なわない様に再度変数を聞き直すプログラムに変更しなければいけません。

GOTO10で始めの方に戻れば、その間が繰り返されて、いちいちRUNは不要

それにしても毎回毎回，ＲＵＮをするのはめんどうだと思いません？

```
10 input a
20 input b
30 c=a+b
40 print c
50 goto 10
```

この GOTO っていうのは何かな？　感じはわかるような気がするけれど…。これはここでは10番に戻れという事なんですって。つまり50番にこの GOTO 10 という命令語を入れておけば，１度ＲＵＮしただけで，10〜40を無限にくり返してくれるって事なのよね。だからいちいちＲＵＮしなくてもいいし，手間がはぶけるわけです。(☞10)

繰り返しを止めるのは SHIFT と BREAK。すぐ続けたいときはOK.の後にCONT

あれおかしい。止まらなくなっちゃった。無限にくり返してくれるのはいいんだけど，止め方がわからないわ!!　ここで，止め方を教えてもらいました。SHIFT を押しながら BREAK のボタンを押せばいいんですって。さあ，これでひと安心。

```
run
? 2
? 3
 5
? 4
? 5
 9
?
Break in 10
Ok.
```

あら？　Ok の右下に . があるけど，これなんだろう？

10

GOTO命令は、実際に使用されている行番号にのみ飛ぶことが出きます。もし行番号の無い所に飛ばすと、

```
GOTO 1000
Undefined label
Ok
▒
```

行き先が無いよ〜と、エラーメッセージを出してくれますよ。

```
Break in 10
Ok.
cont
? 6
? 10
 16
? ■
```

今，GOTO を止めたくて，BREAK したけれど，止めたすぐ次からまた出したい事もあるわよね。Ok にこの．がついている時は，CONT と命令してやればすぐ次の部分からまた表示してくれるそうですよ。．なしのただの Ok ではダメなんですって！　このCONTって何の省略かわかる？　そう Continue の CONT です。Ok の後また RUN をしたって良いわけだけど，RUN の場合は，必ず一番最初の番号からスタートしなければならないわけです。CONT と RUN の違い，わかった？

(☞11)

新しいプログラムを実行する時にはNEWを。今までのプログラムが全部消去

```
new
Ok
■
```

新しいプログラムを実行する時は必ずこのＮＥＷを入れましょう。これを入れることにより，今まで入っていたプログラムが全部消えてしまうわけです。さあ新しいプログラムを作りましょう。

```
10 input a#
20 input b#
30 c#=a#/b#
40 print c#
50 goto 10
run
? 3
? 4
 .75
? 2
? 3
 .6666666666666667
?
Break In 10
Ok.
■
```

11

このちがいがわかりますね。
点が付いている時は、今止まった次の命令から実行させることが可能なのです。もちろんRUNではありません。CONTですよ。

#の記号を使えば、最大の16けたまでの表示をしてくれるのね

#の記号を使ったら……あれ？　さっきわり算のプログラムをした時は8けたまでしか表示されなかったのに，1, 2, 3, 4, …ああ16けたまでちゃんと表示されてるわ。さっき不思議に思ったけれど，ようやく疑問解消!!

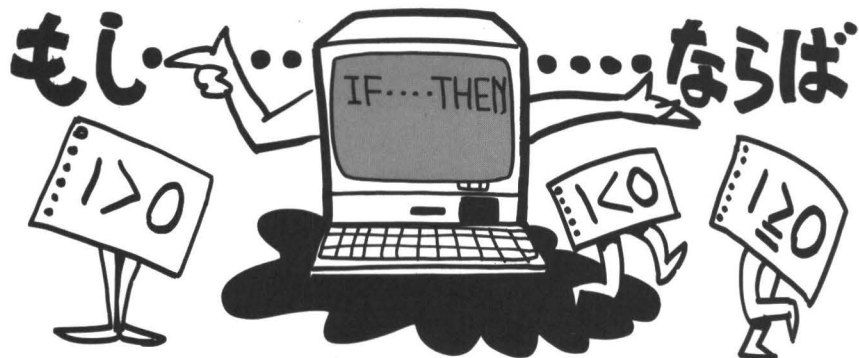
8ケタまでなんて何にも計算できないなあと思っていたけど，これなら安心。国家予算だってなんだって，計算しようと思えばできちゃうわけですね。

もう疲れてきちゃった？　本題はまだまだこれからだけど，ここでちょっと一息つきましょうね。(☞13)

13

倍精度数値#は、有効数字16桁まで表示してくれます。16桁を越える値になると、自動的に浮動小数点形式の表示に変わってしまいます。

```
PRINT 12345678901234
567#[CR]
1.234567890123457D+
16
Ok
▒
```

また倍精度であることを示すために、指数部はDと表示してくれます。同様に単精度（8桁）の時は、Eと表示されます。

コンピュータ言語は簡潔にして各国共通。国際人になるにはコンピュータを!

皆さん外国に興味ありません？　私，とにかく今，外国へ行ってみたいのよね…別に何がしたいというわけじゃないんだけど，こういう時代だから，もう日本とか日本人という枠なんか，全々関係ないですね。

そう考えると語学って重要だけど，それと一緒にコンピュータ知識も必要になってくると思います。なぜって，簡潔で各国共通なもの，人類全般に役立つものなんて，そういくつもありませんよね。今ももちろん必要だけど，そのうち小学生や中学生が学校の授業でコンピュータを教わるという時代もきっと来るんじゃないかしら。つまり，もう国籍も問わずひとつの常識になってしまうと思うんです。とにかく最近，いろいろな意味で，ぜひやっておかなきゃって思うんです。

国際的人間になりたければ，是非コンピュータを！　なんてネ。覚え始めれば，おもしろくなってくるし……まあ，がんばりましょうよ。それにしても，カナダでスキーなんて最高だろうなあ！　行きたいなあ……！

プログラムの内容を全て画面表示するのがLIST。画面から消すのがSHIFTとCLR HOME

間違って画面を消してしまったり，プログラムが見えなくなってしまった場合などにはLISTという命令を使います。そうすると，作っておいたプログラムが全て表示されます。(☞14)

```
list
10 INPUT a
20 INPUT b
30 c=a+b
40 PRINT c
50 GOTO 10
Ok
■
```

それから，画面を消したいなんていう時もありますよね。そんな時は，SHIFT と CLR HOME を同時に押せばいいんです。

ここで注意して欲しいのは，画面からは一度消えても，メモリされているから，LIST すればいつでも，画面に再度現れるという事。

前にＮＥＷという命令をやったけれど，それとの一番大きな相違点がここ!!　ＮＥＷはとにかく消し去っちゃうので，うかつにＮＥＷしちゃうと，後が大変ですよ。どうぞご用心！

画面のクリア（SHIFT ＋ CLR HOME）をやりましたけど，ボタンの下の方に書いてあるHOMEっていうのは何かしらねえ？

これは，カーソル（プログラムの最後に Ok が出たあとに現れる▒）を画面の左上の隅の位置に，移動させる命令です。CLR HOME だけをおすと，HOME を実行してくれるんですって。

この他にも 2" フ　:+ レ などのように，上下書きわけてあるキーがありますよね。2" フ のキーは，ふつうに押せば２が表示されます。フと書きたければ カナ キーをロックしておいてから，2" フ をたたけば OK。そして〃を表示したいときには SHIFT ＋ 2" フ を一緒におせばよいのです。

こういうキーは，SHIFT と一緒におした時に，ボタンの上

14

ＬＩＳＴ CR と命令するとすべてのＬＩＳＴを画面に出してくれます。

LIST 10

と行番号を指定すると、その番号だけ画面に出してくれます。

LIST 20-40

と－で指定すると、その間をすべて画面に出してくれます。

LIST 30-

なら30番から以降をすべて出してくれます。

の方を実行してくれる事になっています。キーボードの右端を見てください。⇦ ⇨ ⇧ ⇩ こんな矢印がありますよね。これはカーソルキーで，矢印の方向にカーソルを動かしてくれます。

ここで少し，大文字，小文字の話をしておきましょうね。この BASIC においては，変数の大文字，小文字は，全て同じものとみなされるんです。プログラムを見てみてください。(☞15)

```
a=1
Ok
A=2
Ok
print a,A
 2             2
Ok
```

HuBASIC では，A と a はまったく同じ変数なのね。

判定文IF～THENを使えば、コンピュータが数の大小を判定してくれます

```
10 input a
20 if a=1 then end
30 goto 10
run
? 4
? 2
? .1
? -1
? 1
Ok
■
```

判定文は IF と THEN とを必ず組み合わせて使わなくてはいけません。上の場合，もしA＝1ならばそこで終わりなさいという命令が20番に入っているんです。Aが1以外だと30番へ行って GOTO 10 の命令により INPUT A をくり返すわけです。

“もし～ならば～しなさい” という判定文なんですよね。THEN の後には，どんな命令を入れてもいいんですって。ちょっといたずらしてみようかしら……。(☞16)

15

→ ← ↑ ↓ カーソルキーを使って、直接実行した行が画面に出ている間は、修正や実行を行なうことができます。

```
PRINT 5+▒
```

16

判定文において、実数の時は良いのですが、変数の時は、THENの前にはかならず1文字分のスペースを入れなければいけません。

```
IF A=B THEN PRINT "E
ND"
```

それ以外のスペースは取り去ってもかまいません。

```
IFA=B THENPRINT"END"
```

```
10 input a
20 if a=1 then print "Yuri"
30 goto 10
run
? 2
? 3
? 1
Yuri
? 1
Yuri
? ■
```

おもしろいなあ。もちろん GOTO で無限にくり返しているので，これを止められるのは BREAK だけです。それでは判定文を使って大小比較文を作ってみましょう。

```
10 input a
20 input b
30 if a=b then print "a wa b to onaji"
40 if a<b then print "a wa b yori chiisai"
50 if a>b then print "a wa b yori ookii"
60 goto 10
```

IF…THEN を使うとこんな事ができますよ。さあＲＵＮさせてみましょうね。

```
run
? 2
? 3
a wa b yori chiisai
? ■
```

こんな具合に数の大小を，コンピュータが判断してくれます。このように IF…THEN 文の活用範囲は広いんですよ。

この場合，記号で表わすとすれば，不等号を使うことになるわけですが，この不等号は算数と同じように＜，＞と書きます。読み方は……。

＜　less than

＞　greater than

と呼ぶのだそうです，意味は，わかりますよね。

カナKEYをロックして、カナモードにして、日本文でも数の大小を表現しましょう

このプログラムをもう少し見やすくするために，カナモードで書いてみましょう。カナ・モードにするには **カナ** KEYをロックするだけでＯＫ！(☞17)

```
10 input a
20 input b
30 if a=b then print "a ﾊ b ﾄ ｵﾅｼﾞ"
40 if a<b then print "a ﾊ b ﾖﾘ ﾁｲｻｲ"
50 if a>b then print "a ﾊ b ﾖﾘ ｵｵｷｲ"
60 goto 10
run
? 1
? 1
a ﾊ b ﾄ ｵﾅｼﾞ
? 2
? 3
a ﾊ b ﾖﾘ ﾁｲｻｲ
?
Break In 10
Ok.
■
```

やっぱりカタカナの方が見やすいなあ。

A=B、A<Bなら残りは必然的にA>Bで"AはBより大きい"ということになります。

さて，さっきの IF…THEN 文のプログラムにもどりますよ。

30，40，50と３つも IF…THEN 文がならんでいますけど，Ａ＝ＢじゃないとしてＡ＜Ｂでもなければ，あとは全部Ａ＞Ｂになるのは，わかりきっていると思いません？

だから50の IF A>B THEN PRINT "AハBヨリオオキイ" っていうのは余分なんじゃないかしら。つまり50番には，ただ "AハBヨリオオキイ" と書けとだけ命令すれば，いいんじゃないかしら。何か先生がおかしな顔して見てますけど，私，ちょっとプログラムを書きなおしてみますネ！

17
カナを使った後は、かならずカナロックを解除しておいてください。

```
list
10 INPUT a
20 INPUT b
30 IF a=b THEN PRINT "a ﾊ b ﾄ ｵﾅｼﾞ"
40 IF a<b THEN PRINT "a ﾊ b ﾖﾘ ﾁｲｻｲ"
50 PRINT "a ﾊ b ﾖﾘ ｵｵｷｲ"
60 GOTO 10
Ok.
■
```

これで完璧だと思いません？

A>Bを省略したはいいけれど、無条件に"AはBより大きい"が出ては大変！

じゃあ早速実行させてみますよ！

```
run
? 1
? 1
a ﾊ b ﾄ ｵﾅｼﾞ
a ﾊ b ﾖﾘ ｵｵｷｲ
? 3
? 2
a ﾊ b ﾖﾘ ｵｵｷｲ
? 2
? 3
a ﾊ b ﾖﾘ ﾁｲｻｲ
a ﾊ b ﾖﾘ ｵｵｷｲ
? ■
```

ああ！　これだからダメなのよね。やっぱりもっと慎重にならなきゃネ。そうです，よく考えてみるとおかしいんですよネ。AがBより大きい時は問題ないけれど，30番，40番のどちらかにあてはまって，表示された後で，自動的に50番も実行しちゃう事になるんです。

だから上のような表示になっちゃったわけ。

A=B、A<Bとなるたびに、GOTO10でINPUTの条件を確認する

でもこのアイディアはちょっと捨てがたい気がするんだけれど……もうちょっとよく考えてみましょうよ。何かいい方法があるんじゃないかしら……ここで悩んだあげくに思いついたのが次のプログラム。よく見て下さいネ！

```
list
10 INPUT a
20 INPUT b
30 IF a=b THEN PRINT "a ハ b ト オナシ゛":GOTO 10
40 IF a<b THEN PRINT "a ハ b ヨリ チイサイ":GOTO 10
50 PRINT "a ハ b ヨリ オオキイ"
60 GOTO 10
Ok.
```

30番と40番のあとにコロンを打って GOTO 10 をつけ加えたんです。コロン［:＊ケ］とは，まだ文がつづきますよという記号です。覚えておいて下さい。

こうしておけば，さっきのような混乱はなく，一応整理されてくると思うんだけど，どうかな？　余分な命令は省いて，できるだけ簡略にした方が難しいプログラムになる程いいに決まってますよねえ！ではちょっと心配だけど，ＲＵＮ!!

```
run
? 1
? 1
a ハ b ト オナシ゛
? 1
? 2
a ハ b ヨリ チイサイ
? 2
? 1
a ハ b ヨリ オオキイ
?
```

ELSE命令を使うと、3つの命令文を1つにまとめることができるのです

一応，すんなり行ってるみたいですねえ，良かった！　でも先生は，これでも完璧ではないとおっしゃっていますよ。

ここで大変便利な命令を，教えて下さいました。ELSE という命令なんです！　これを使うと，なんと30番40番50番の３つの命令文を１つにしてしまえるんです。本当かなあって思うでしょうねえ。さあプログラムを見て下さい。(☞18)

18
１行に入る命令の長さは最大255文字までです。又行番号はプログラムの先頭に０から65534までの番号を付けることによって、若い番号順に実行してくれるのです。

```
list
10 INPUT a
20 INPUT b
30 IF a=b THEN PRINT "a ハ b ト オナジ" ELSE
 IF a<b THEN PRINT "a ハ b ヨリ チイサイ" ELSE
PRINT "a ハ b ヨリ オオキイ"
60 GOTO 10
Ok
■
```

プログラム上で消したい行があるときは、その行番号とCRを順に押すのです

40番，50番は，上のプログラムでは，もう必要がないので，消しちゃっていいわけです。消し方は，4 0 CR，5 0 CR といった感じでいいそうです。こうすると簡単になるでしょ？

ELSE というのは，“さもなければ”という意味ですよネ。IF…THEN…ELSE このつながりが重要！　これらは切り離して使えるものではないのです。(☞19)

だんだんコンピュータらしくなってきて判定をしてくれるようになりました。この先，どんな事ができるようになるのかワクワク！　でも，まだまだ完全に私の上をいっているコンピュータ。少し複雑な気分ですねぇ。

ところで IF……THEN……ELSE の命令文なんかとってみると，まったく英語と同じだし，意外に親しみやすいのよね。

数字だけがぎっしり並んでいるように思えたんだけど……それに１＋１＝２みたいに，決まりきったような計算しか判断してくれないと思っていたのに，意外に幅があるんです。

そういう意味では，わりと人間的だともいえるんじゃないかしら…。そんなわけで，私の中では，コンピュータに対する認識が少しずつ変化してきているんだけど，皆さんはどうかな？上手に使いこなせるようになれば，もっと愛着もわいて，自分の分身みたいな気がするんじゃないかしら。

19
ＥＬＳＥを使うことによって、ＩＦ文の後に何個でも命令をつづけることができますが、前にも言ったように、最大255文字までですよ。

RNDはランダム、そうです、でたらめな数つまり乱数を出すための命令なのです

それでは，判定文を使ってゲームのプログラムを作ってみましょう。ここで**RND（　）**という新たな記号が出てきます。これは**乱数**と呼びます。乱数というのは，その字の通りでたらめな数。ですから（　）の中には，どんな整数が入ってもいいけれど，普通はRND(1)とするそうですよ。ダイレクト・モードでやってみましょう。(☞20)

```
print rnd(1)
 .98755252
Ok
print rnd(1)
 .57657397
Ok
print rnd(1)
 .89712572
Ok
■
```

このように，小数点以下８けたの適当な数を出してくれます。

20
RND(X)乱数の引数Xはどんな数字を入れてもかまいません。単なるダミーです。RND(1)だけを実行しますと、０以上１未満の乱数を発生してくれます。

小数を整数になおす操作は、10^nをかけて小数点を移動させ、INTで小数を切り捨て!

でも小数点以下の数しか出なければ，あんまり利用できないんじゃないかしら？　すると先生が「算数を思い出してごらんなさい。小数を整数にする事はできるでしょ」とおっしゃいました。

難しい事を言うなぁ。かけ算を使うっていう事かナ？　確かに，0.123456に10をかけたら1.23456になるわねぇ。でも小数点以下の数があるし，これじゃ整数といえませんね。ここで整数だけ残して，小数点以下を全部切り捨ててしまう命令があるんですよ。

これがＩＮＴ。**インテジャー**(integer)と呼びます。さっきのＲＮＤ（　）乱数も，このインテジャーもコンピュータでは**関数**と呼ばれます。(☞21)

```
print int 1.2
Syntax error
Ok
print int(1.2)
 1
Ok
■
```

INTは( )と併用するもの。( )内にRND(1)を入れて1つの命令文にまとめて……

このようにＩＮＴを使う時は，必ず数値を（　）で閉じなくちゃダメなんですよ。これもルールだと思って下さい。

```
print int(5.36
Syntax error
Ok
■
```

ほらね，（　）を閉じてないから，こんな事になっちゃったでしょ。

21
INTEGERとは，「整数の」という意味ですね。その他にも「必要な」とか，「完全な」などという意味もあるのですよ。「必要な整数を出してくださいね」と覚えましょう。

```
print rnd(1)*10
 6.8445432
Ok
print int(6.8445432)
 6
Ok
■
```

こうなりますよね。

でもいちいち小数点以下を切りすてる命令をするのは，ちょっとめんどうよねえ。これを一つの命令文には，できないのかしら……？

```
print int(rnd(1)*10)
 4
Ok
■
```

こうすれば一つの命令文でいいわけよねぇ。(　) が多くなるから，気をつけてネ！

GOTO10を打ち込むと、小数を切り捨てた乱数がいくつも…、0も含まれます

```
10 print int(rnd(1)*10)
20 goto 10
```

先生がこんなプログラムを書きました。「わかりますか？」ですって…はい，わかりますよ。これは，さっきの乱数をくり返しなさいというプログラムですね。さあそれじゃ実行してみましょう。

```
run
 0
 3
 2
 1
 6
 7
 9
 5
 7
Break in 10
Ok.
■
```

こんな風に10までの乱数を出してくれるんだわ。でも先生，なぜ0が出てるの？

```
print .099*10
 .99
Ok
■
```

この計算は理解できますね。0.099 に10をかけたって 0.99 。そして小数点以下を INT で切り捨ててしまうので0になるのです。

サイコロの目の乱数を出してみましょう! INT(RND(1)*6)に1を加えて……

じゃあちょっと別のプログラムを作ってみましょうか。

```
10 print int(rnd(1)*6)
20 goto 10
```

6までの乱数を出しなさいというプログラムですね。これはよくゲームに使われますよ。なぜ6にしたかわかります？　何か思い出す物はありませんか？　そう！　サイコロです。じゃあサイコロの目を出してみましょう。

```
run
 1
 4
 5
 0
 4
 3
 0
 2
Break In 10
Ok.
■
```

となりました。これがサイコロの目なのかしら？　よく見て下さい。変なサイコロねぇ！　先生，サイコロの目に0なんかないでしょ。それに6がないですよ。困りましたねぇ，どうしたらいいのかしら。つまり，1つずつ数が大きくなればいいわけよねえ。じゃあ,もとのプログラムに1をたしてみましょうよ。

```
10 print int(rnd(1)*6)+1
20 goto 10
run
 1
 5
 6
 6
 1
 2
 2
 3
 4
 3
 6
 3
Break In 10
Ok.
■
```

やった！　大成功。これでサイコロの数当てゲームなんかやって遊べますよ。これを応用すればトランプゲームでもできちゃうわけよねえ。どう？　やっと，それらしくなってきたでしょ。

忘れないで欲しい事は，コンピュータは数を数える時は0から数え出すのだという事です。だから PRINT INT (RND(1)＊6)+1 の＋1 に意味が出てくるわけね。(☞22)

サイの目乱数Aに対して、BをINPUTして条件を与えれば、サイコロ数当てゲーム

それじゃ，いよいよサイコロの数当てゲームを作ってみましょう。

```
10 a=int(rnd(1)*6)+1
20 input b
30 if a=b then print "ｱﾀﾘ !"
40 if a<>b then print "ﾊｽﾞﾚ.."
50 goto 10
■
```

これでプログラムができましたね。40番のところに見慣れぬ記号が出てきましたが，これは A ≠ B と同じ意味。“ノットイコール”と読みます。(SHIFT ＋ [，＜、ネ] ＋ [・＞。ル]) と押してみてください。(☞23)

22
A=INT(RND(1)*6)+1
A=INT(RND(1)*6+1)
A=INT(6*RND(1)+1)
この様に乱数を発生させたい数は、前後どちらに置いても良いのですよ。

23
ノットイクオールは、
<>　><
どちらでも良いですよ。

このプログラム，間違いないですよねえ？　どう思う？　それじゃあ，とにかくＲＵＮ！

```
run
? 4
ﾊｽﾞﾚ..
? 5
ﾊｽﾞﾚ..
? 6
ｱﾀﾘ !
?
Break In 20
Ok.
■
```

数当てゲームは、A=Bか否かを問題にしてるのだからELSEを使って合理化

うまくいきました。別に間違っていないけど……。

ここで何か気づきません？　何も判定文を２つにする必要はないでしょ？　Ａ＝Ｂ以外は全部 ハズレ になっちゃうんだから一つの判定文になるわよね！　さて，どうするんだっけ？

```
list
10 a=INT(RND(1)*6)+1
20 INPUT b
30 IF a=b THEN PRINT "ｱﾀﾘ !" ELSE PRINT "ﾊｽﾞﾚ.."
50 GOTO 10
Ok
■
```

これでいいのよネ，40番はもう必要ありません。

```
run
? 1
ﾊｽﾞﾚ..
? 2
ｱﾀﾘ !
? 3
ﾊｽﾞﾚ..
?
Break in 20
Ok.
■
```

きちんとできました。先生はただ＜　＞の説明がしたかったので最初あのプログラムを作ったんですって。

サイの目なのに6より大きい数が混入したら、全てハズレになります

ここで，もしサイコロの目だっていう事を忘れて，7とか9とか入力したら，どういう判定になるのかしら？

```
run
? 7
ﾊｽﾞﾚ..
? 9
ﾊｽﾞﾚ..
? 11
ﾊｽﾞﾚ..
? ■
```

全部ハズレではサイコロゲームにならないので、B<1、B>6をやり直しさせます

これじゃ全部はずれになっちゃってぜんぜんゲームらしくないじゃない！　そこで，さらに条件をつけて，判定を細かくしてみましょう。

```
list
10 a=INT(RND(1)*6)+1
20 INPUT b
25 IF b<1 THEN GOTO 20
26 IF b>6 THEN GOTO 20
30 IF a=b THEN PRINT "ｱﾀﾘ !" ELSE PRINT
"ﾊｽﾞﾚ.."
50 GOTO 10
Ok
■
```

25，26番に入れた命令により，ゲームらしくなったはずですね。つまり1～6以外の数が出てきたら，コンピュータは判定せず，さらに聞きかえしてくれるわけです。(☞24)

24
コンピュータのプログラムを理解するためには、とにかくコンピュータと遊んでみることが一番ですね。遊びの中から不合理な点を見付け出し、それを修正して行くのです。それができればもうあなたはプログラマーなのですよ。

1より小さい数は"小数点以下だけを取り出しなさい"というFRACTIONで処理!

```
run
? 1.5
ハズ゛レ..
? 3.5
ハズ゛レ..
? ■
```

あれ？　1より小さい数字と6より大きい数字は，はね返してくれるけど，1.5がハズレなんておかしいわ。だってサイコロにそんな目ないわよね。これは，ちょっとめんどうな事になりましたねえ。

ここで先生は悩んだすえ，FRACTIONという命令を教えてくれました。これはどういう命令かと言いますと，"小数点以下だけをとり出しなさい"という命令なんだそうです。(☞25)

```
print frac(1.15)
 .15
Ok.
print frac(5)
 0
Ok.
print frac(3.451)
 .451
Ok.
■
```

このように小数点以下のみを表示し，整数の場合は0となるわけです。これを上手に使って，もっと正確なプログラムを作ってみましょう！

25

小数点以下を取り出すのは、FRAC(X)、小数点以上を取り出すのはFIX(X)

```
PRINT FRAC(2.35)
 .35
Ok
PRINT FIX(2.35)
 2
Ok
▒
```

```
list
10 a=INT(RND(1)*6)+1
20 INPUT b
22 IF FRAC(b)<>0 THEN GOTO 20
25 IF b<1 THEN GOTO 20
26 IF b>6 THEN GOTO 20
30 IF a=b THEN PRINT "ｱﾀﾘ !" ELSE PRINT
"ﾊｽﾞﾚ.."
50 GOTO 10
Ok
■
```

サイの目が"1より小さい"が"6より大きい"この2つの判定文はORでつながっています

こんな風にFRACTIONを使ってプログラムを作り直しました。乱数を出させた後，小数点以下のあるなしをふるいにかけるわけです。もし，小数点以下があれば，また20番へ戻ります。先生いわく，もう少しスマートな形にしようという事で，ちょっとプログラムを書き直してみました。(☞26)

```
list
10 a=INT(RND(1)*6)+1
20 INPUT b
22 IF FRAC(b)<>0 THEN GOTO 20
25 IF b<1 OR b>6 THEN 20
26 IF b>6 THEN GOTO 20
30 IF a=b THEN PRINT "ｱﾀﾘ !" ELSE PRINT
"ﾊｽﾞﾚ.."
50 GOTO 10
Ok
■
```

さっきのプログラムと比較して，25番が違っていますね。26番の内容もつなげてしまったわけです。そのために使ったのが論理演算子のOR。この呼び名は覚えておいてネ！

IF~THEN~GOTOの文では、GOTOがなくてもコンピュータにはわかるの

ORは英語のorとまったく同じ使い方。さらにその後，THEN 20となって，GOTOがきえていますが，これはもう入れなくても大丈夫なんです。IF……THENのあとのGOTOは，コンピュータがちゃんと記憶して憶えていてくれるんですよ。

26
プログラムは、できるかぎり短く簡単にまとめるのがテクニックです。HuBASICの数々のすぐれた命令を駆使して、スマートなプログラムを完成させましょう。

どうです？　スマートにしかも正確になってきましたよ。命令文がゴチャゴチャしているのは間違いのもと。できるだけ簡潔にね。

さあ，また今のプログラムに戻りましょう。先生は，このプログラムをまだ変えたいみたいですよ。もっと簡単にできるんですって。本当かな？　でも，また新しい命令が出てくるのかしら？　もう大分出てきたわよねえ。

ORの仲間にANDがあります。ANDは両方の条件を充す演算子なのです

とにかく，続きを教わりましょう。今度は **AND** を使うんですって。なんでしょうねぇ？　これもORの仲間で**論理演算子**なんだそうですよ。

```
LIST
10 a=INT(RND(1)*6)+1
20 INPUT b
22 IF FRAC(b)<>0 THEN 20
25 IF b>1 AND b<6 THEN 30
26 GOTO 20
30 IF a=b THEN PRINT "ｱﾀﾘ !" ELSE PRINT
"ﾊｽﾞﾚ.."
50 GOTO 10
Ok
■
```

```
run
? 1
? 2
ﾊｽﾞﾚ..
? 3
ﾊｽﾞﾚ..
? 4
ﾊｽﾞﾚ..
? 5
ﾊｽﾞﾚ..
? 6
? ■
```

ANDの使い方がよくわからないけれど，このプログラムはいったい何なのかしら。おかしいなぁ？　どうして1と6はうけつけないのかしら……。じゃあ，ちょっと整理してみましょうネ。

B<1 OR B>6の条件の反対の条件をANDで構成させるためには1≦B≦6にして……

```
A=B    A ト B ハ ヒトシイ。
A<B    A ハ B ヨリ チイサイ。
A>B    A ハ B ヨリ オオキイ。
A<>B   A ト B ハ ヒトシクナイ。
A<=B   A ハ B ト オナジ゛カ B ヨリ チイサイ。
A>=B   A ハ B ト オナジ゛カ B ヨリ オオキイ。
```

さっきORのところで，B＜1　OR　B＞6とやったけど，あの場合は1と6を含まないのだからあれでよかったわけです。今度は1≦B≦6としたいわけだからコンピュータの表示方法で，B＞＝1　AND　B＜＝6　としなきゃいけないわけよネ。ANDとは2つの条件両方を満足させねばならないとき使う演算子です。それじゃ，25番を直しましょう。

```
25 if b>=1 and b<=6 then 30
```

はい，これでやっとできあがりネ！　もちろん22と25をELSEでつなぐ事も可能ですよ。(☞27)

```
22 if frac(b)<>0 then 20 else if b>=1
   and b<=6 then 30 else 20
```

27

論理演算子と呼ばれる物の中にはAND、ORの他に次の命令があります。

NOT　否定
XOR　排他的論理和
IMP　包含
EQV　同値
AND　論理積
OR　論理和

一般にはこのANDとORが良く使われ、他の論理演算子は、あまり使用されませんのでここでは省略いたします。

メッセージ文の入れ方は;と変数を忘れずにつけることなのです

ここで，メッセージの入れ方を教わりましょう。ただの〔**?**〕じゃ何を聞いているのかわからないので，ここでは「あなたの予想は」というメッセージを入れてみましょうネ。

```
list
10  a=INT(RND(1)*6)+1
20  INPUT  "ｱﾅﾀﾉ ﾖｿｳ ﾊ ";b
22  IF FRAC(b)<>0 THEN 20 ELSE IF b>=1
    AND b<=6 THEN 30 ELSE 20
30  IF a=b THEN PRINT  "ｱﾀﾘ !" ELSE PRINT
"ﾊｽﾞﾚ..":GOTO 20
50  GOTO 10
Ok
■
```

```
run
ｱﾅﾀﾉ ﾖｿｳ ﾊ ? 1
ｱﾀﾘ !
ｱﾅﾀﾉ ﾖｿｳ ﾊ ? 2
ﾊｽﾞﾚ..
ｱﾅﾀﾉ ﾖｿｳ ﾊ ? 3
ﾊｽﾞﾚ..
ｱﾅﾀﾉ ﾖｿｳ ﾊ ? 5
ｱﾀﾘ !
ｱﾅﾀﾉ ﾖｿｳ ﾊ ? ■
```

ここで絶対忘れてはいけないのが；(**セミコロン**)。そして，そのあと変数を忘れずにつけて下さいネ。そうすると見やすいでしょう。

PRINT 文の内容も，カタ仮名にするとか，いろいろ変えられるわけです。

さあ，今せっかく作ったゲームを記録しておきたいんですが，どうするんでしょうか？(☞28)

28

INPUT文の中にも、PRINT文同様にメッセージを書くことができるのです。

ただ、

```
PRINT"ｱﾅﾀﾉﾅﾏｴ ﾊ ";A$
;" ｻﾝﾃﾞｽ"
```

この様に、プリント文の中に文字変数を入れることはできても、INPUT文の中には入れることはできません。、

```
10 INPUT "ｱﾅﾀﾉﾅﾏｴ ﾊ
";A$
20 PRINT "ｱﾅﾀﾉﾅﾏｴ ﾊ
";A$;" ｻﾝﾃﾞｽ"
```

この様に書くとOKです。

プログラムをカセットに記録したい時はSAVE文を使ってタイトルを入れます

ここで出てくるのが，**SAVE** という命令です。記録したいときは，この SAVE を押してタイトルを入れます。「**カズアテゲーム**」としましょう。

```
save"ｶｽﾞｱﾃ ｹﾞｰﾑ"
Writing"ｶｽﾞｱﾃ ｹﾞｰﾑ     .  "
Ok
▒
```

```
save "ｶｽﾞｱﾃ ｹﾞｰﾑ"
Out of tape
Ok
■
```

カセットテープが入っていないと、こんなエラーメッセージが出ますよ

本当にカセット記録されたかどうかを確認するためにはLOAD命令を使います

では，巻き戻して，本当に記録されたかどうか調べてみましょう。ここで LOAD？ という命令を使います。

```
load?
Found  "ｶｽﾞｱﾃ ｹﾞｰﾑ     .  "
Ok
▒
```

これでテープの内容と，プログラムの内容が一致していることがわかりました。

それでは，このゲームをもっとゲームらしくするために10回やって統計をとりましょう。

そのために数を数えるためのプログラムを作ってみます。

```
10 a=a+1
20 print a
30 if a=10 then 50
40 goto 10
50 print "end"
run
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
end
Ok
■
```

BASICではRUN命令の実行は全て変数0から始まるのでよく憶えておいてネ!

BASICでは，RUN命令を実行する時は，すべての変数は0から始まります。

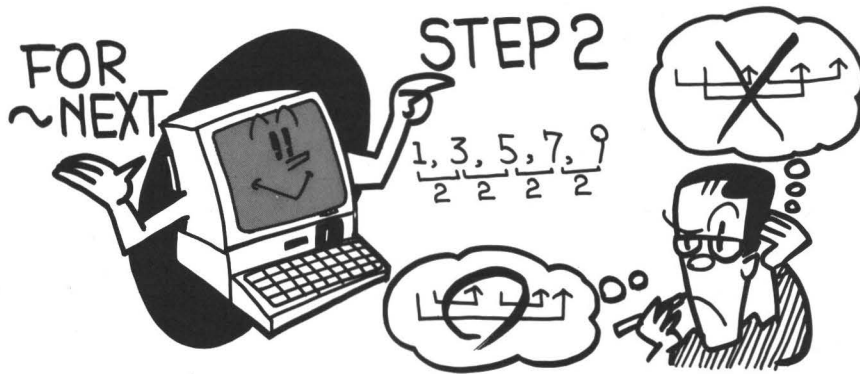
例えば，このプログラムを理解する上で大切なのは10番です。A＝A＋1では，まず右側のAに0が入りますね。そうすると左側のAは1になります。つまり1＝0＋1という事です。

ここでは，右と左のAはそれぞれ別の器(うつわ)だと考えて下さい。いつでも右の方のAに先に数字が入り，左側のAが1づつ増えていく事になるのです。

```
 1=0+1
 2=1+1
 3=2+1
 4=3+1
 5=4+1
 6=5+1
 7=6+1
 8=7+1
 9=8+1
10=9+1
```

```
 A=A+1
```

これが全部A＝A＋1の式の中で行われるわけです。これで数の数え方のプログラムはいいですね。

これを利用して，ゲームをくり返しやれば，もっと楽しめます。

A=A+1の代わりにFOR~NEXT文でAを順次変化させれば、もっと楽に…

ところが，これよりもっと簡単なプログラムで，今とまったく同じことができるんですって？

```
new
Ok
10  for a=1  to 10
20  print a
30  next a
40  print "end"
run
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
end
Ok
■
```

ここで，FOR…NEXT という命令を使いました。1～10までの間で，さっきのプログラムとまったく同じ事をこの命令はやってくれています。

FOR~NEXTを複雑に組み合わせて級数の行列を構成することもできます

FOR…NEXT は大変便利な命令で，いろいろな使い方ができます。たとえば，次のような事もできるんですよ。

```
10 for a=1 to 3        ①
20 for b=1 to 3
30 print a,b
40 next b
50 next a
run
 1             1
 1             2
 1             3
 2             1
 2             2
 2             3
 3             1
 3             2
 3             3
Ok
■
```

このように，複数組み合わせて使えば行列だってできちゃうんですよ。ただ使い方は気をつけてください。ちょっとしたルールがあるみたいですよ。

```
10 for a=1 to 3        ②
20 for b=1 to 3
30 next b
40 for c=1 to 3
50 next c
60 next a
■
```

```
10 for a=1 to 3        ③
20 for b=1 to 3
30 next a
40 next b
run
For without next in 10
Ok
■
```

FOR〜NEXTのループがいくつあってもいいけど、ループがクロスしてはダメ!

FOR…NEXT がいくつできても，①，②のように外側から順番にループができていれば大丈夫（㊟ [FOR NEXT の [をループと呼ぶそうですよ）。

要するに，プログラムの中で FOR…NEXT のループがクロスしてなければ良いわけです。

③のようなまちがったプログラムをすると，すぐに，

```
Next without for
For without next
```

というような ERROR 表示が出てきます。実行する順番は，行番号の小さなものからですよ。(☞29)

だから，10番が FOR A=1 TO 3 だったらペアの NEXT A つまり50番をみつけて，実行するわけです。わかりました？

STEP命令は数の増減の幅を指定する命令。STEP1なら省略できるのです

ここで，もう1つ補足。これまで STEP という命令には触れてませんが，今までの命令文は，実はすべて STEP が1の命令なんだそうですよ。(☞30)

STEP とは，変動の幅のようなものなんですって。さっき作ったプログラムは RUN すると1，2，3，4と1つずつ大きくなっていったでしょ。これが STEP 1 という事で，この場合だけは省略できるそうです。ところが，STEP 2 とか STEP 3 になるとどうなるか，実際に出力させてみますね。

29

FORからNEXTまでの間にある文が、指定された回数繰り返し実行されます。FOR文の変数には、始めに初期値が入れられ、以後この文が実行されるたびに、1ケずつ増えて行き、最終値になったら終了いたします。

FOR I=1 TO 10
変数 初期値 最終値

なお、FORとNEXTは、かならずペアで使われます。

30

FOR I=1 TO 10 STEP 2
変数 初期値 最終値 増分

FOR文の変数には、始めに初期値が入れられ、以後この文が実行されるたびに増分ずつ増えて行くのです。もちろん、最終値になったら終了ですよ。

```
10 for a=1 to 9 step 2
20 print a
30 next a
40 end
run
 1
 3
 5
 7
 9
Ok
■
```

こんな風になるんですね。すごく便利だと思いません。

STEP命令を使って九九の計算をさせるにはA＊BのA、Bをステップさせます

ということは……応用で九九（くく）の計算なんていう事も可能なわけかしら……。

うーん，ちょっと難しいプログラムになっちゃうかな？　わからなくなったら先生教えてくださいネ！

```
10 for a=0 to 9
20 for b=0 to 9
30 print a*b
40 next b
50 print
60 next a
70 end
■
```

```
run
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
```

0
1
2
3
4
5
6
7
8
9

0
2
4
6
8
10
12
14
16
18

0
3
6
9
12
15
18
21
24
27

0
4
8
12
16
20
24
28
32
36

0
5
10
15
20
25
30
35
40
45

```
 0
 6
 12
 18
 24
 30
 36
 42
 48
 54

 0
 7
 14
 21
 28
 35
 42
 49
 56
 63

 0
 8
 16
 24
 32
 40
 48
 56
 64
 72

 0
 9
 18
 27
 36
 45
 54
 63
 72
 81

Ok
```

九九の計算を横に表示するには、A＊B;のように最後に;をつけます

どうやらプログラムは正しかったみたい。でも，RUN したとたん，たて一列に数字がどんどん出てきて，これじゃとても見にくいですよね。先生何か良い方法はないでしょうか？

えっ！　先生は30番の最後にセミコロンをつけるだけで大丈夫なんて言ってますけど，ほんとでしょうか。試してみましょうね。

まずは……，

```
list
10 FOR a=0 TO 9
20 FOR b=0 TO 9
30 PRINT a*b;
40 NEXT b
50 PRINT
60 NEXT a
70 END
Ok
■
```

直すのはこれだけでいいのかしら……。

```
run
 0  0  0  0  0  0  0  0  0  0
 0  1  2  3  4  5  6  7  8  9
 0  2  4  6  8  10  12  14  16  18
 0  3  6  9  12  15  18  21  24  27
 0  4  8  12  16  20  24  28  32  36
 0  5  10  15  20  25  30  35  40  45
 0  6  12  18  24  30  36  42  48  54
 0  7  14  21  28  35  42  49  56  63
 0  8  16  24  32  40  48  56  64  72
 0  9  18  27  36  45  54  63  72  81
Ok
■
```

やった!!　セミコロンを使えば，横に並べて表示できるのね！　ちょっとした違いで，画面は全く変ってしまいました。大感激！(☞31)

31

セパレータ

: 　マルチステートメント時使用
; 　変数のつなぎに使用
, 　変数のつなぎに使用

```
PRINT 1;2
␣1␣␣2
Ok
PRINT 1,2
␣1␣␣␣␣␣␣␣␣␣2
Ok
PRINT "A";"B"
AB
Ok
PRINT "A","B"
A␣␣␣␣␣␣␣␣␣B
Ok
▒
```

数字の前には、かならず1文字あけられますので注意してください。

九九の計算プログラムは要するにFOR～NEXT文のかけ合わせなのです

九九の計算のプログラム，理解してもらえたかしら。ここでもう少し具体的に説明してみましょうね。

まず行番号10で１回目にＡ＝０，そして次の行番号20でも１回目のＢを０とします。

30番では，Ａ＊Ｂの計算結果を表示し，次の行番号へと行くわけです。

でも，40番には２番目のFOR～NEXTループのNEXTがくるので，20番に戻る事になり，Ｂの値は増加されＢ＝１となります。それから30番を実行します。このようなくりかえしをＢが９以上になるまで実行します。

Ｂが９までまわると，行番号50のPRINT命令により表示の行換えを行ない，次の60番のNEXTで行番号10に戻ります。そしてＡが１つ増えＡ=1となります。同じような事をＡが９になるまでくり返します。

```
a=0                          a=1
 a   b        a*b             a   b        a*b
 0   0         0              1   0         0
 0   1         0              1   1         1
 0   2         0              1   2         2
 0   3         0              1   3         3
 0   4         0              1   4         4
 0   5         0              1   5         5
 0   6         0              1   6         6
 0   7         0              1   7         7
 0   8         0              1   8         8
 0   9         0              1   9         9
```

つまり，この表のような事が，Ａが９になるまで実行されるのですね。

こんな風に順を追ってみていけば，たいして難しくもないでしょう。どうですか？

FOR~NEXT文でタイマーを作ることもできます。START~END間は1秒

FOR…NEXT 文のおもしろさがわかってもらえたかしら。それでは，ここでちょっと変わった FOR…NEXT 文の使い方を見てみましょうね。

実は，タイマーとして利用できるんです。

```
10 print "start"
20 for a=1 to 1000
30 next a
40 print "end"
run
start
end
Ok
■
```

START から END まで，毎回きまって1秒かかることになります。

```
10 for a=1 to 3
20 for b=1 to 3
30 print a,b
40 next
50 next
```

このように，NEXT に変数をつけなくても，コンピュータはちゃんとどの変数をつけるかを知っています。かえって変数をつけない方が，実行速度ははやくなるんですよ。(☞32)

32
FOR~NEXTを利用したタイマーは、ゲーム等で良く使われます。値を変えていろいろ試してみて下さい。

SAVEした"カズアテゲーム"をFOR〜NEXT文を使って書き換えてみます

それでは，前に作ったゲームに応用してみましょう。じゃあ，さき程 SAVE したゲームのテープを LOAD してみましょうね。

```
load
Found   "ｶｽﾞｱﾃ ｹﾞｰﾑ    .    "
Ok
list
10 a=INT(RND(1)*6)+1
20 INPUT "ｱﾅﾀﾉ ﾖｿｳ ﾊ ";b
22 IF FRAC(b)<>0 THEN 20 ELSE IF b>=1
   AND b<=6 THEN 30 ELSE 20
30 IF a=b THEN PRINT "ｱﾀﾘ !" ELSE PRINT
"ﾊｽﾞﾚ..":GOTO 20
50 GOTO 10
Ok
■
```

このように書きかえてみましょう。

```
list
5 FOR i=1 TO 5
10 a=INT(RND(1)*6)+1
20 INPUT "ｱﾅﾀﾉ ﾖｿｳ ﾊ ";b
22 IF FRAC(b)<>0 THEN 20 ELSE IF b>=1
   AND b<=6 THEN 30 ELSE 20
30 IF a=b THEN PRINT "ｱﾀﾘ !" ELSE PRINT
"ﾊｽﾞﾚ..":GOTO 20
50 NEXT i
Ok
■
```

アタリの回数をPRINT"アタリノカイスウ"ではじき出すこともできます

さらに次のように回数を指定できますね。

```
list
5 FOR i=1 TO 5
10 a=INT(RND(1)*6)+1
20 INPUT "ｱﾅﾀﾉ ﾖｿｳ ﾊ ";b
22 IF FRAC(b)<>0 THEN 20 ELSE IF b>=1
   AND b<=6 THEN 30 ELSE 20
30 IF a=b THEN PRINT "ｱﾀﾘ !":x=x+1:GOTO
50
40 PRINT "ﾊｽﾞﾚ.."
50 NEXT i
60 PRINT "ｱﾀﾘﾉ ｶｲｽｳ=";x
Ok
■
```

そうすると……，

```
run
ｱﾅﾀﾉ ﾖｿｳ ﾊ ? 2
ｱﾀﾘ !
ｱﾅﾀﾉ ﾖｿｳ ﾊ ? 5
ﾊｽﾞﾚ..
ｱﾅﾀﾉ ﾖｿｳ ﾊ ? 6
ｱﾀﾘ !
ｱﾅﾀﾉ ﾖｿｳ ﾊ ? 3
ﾊｽﾞﾚ..
ｱﾅﾀﾉ ﾖｿｳ ﾊ ? 4
ﾊｽﾞﾚ..
ｱﾀﾘﾉ ｶｲｽｳ= 2
Ok
■
```

5回のトータルであたりの回数を出してくれます。

RENUMを使うと10番からキッチリ10番おきに行番号を改めてくれます

さて次に，行番号を RENUM [CR] を使って10番から10番おきにきれいにそろえてみましょう。

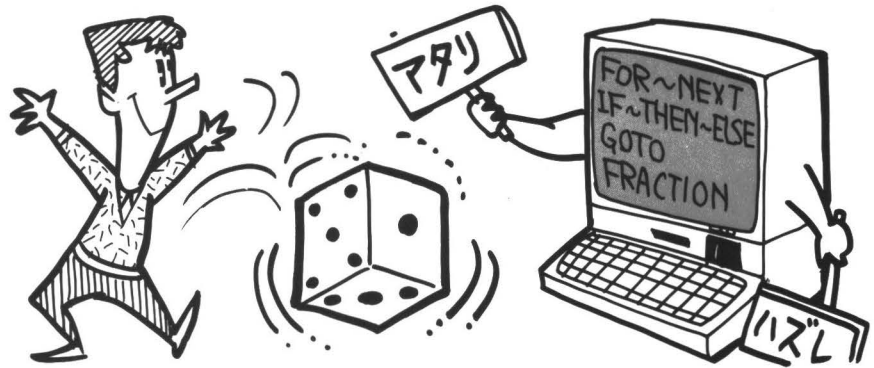
```
list
5 FOR i=1 TO 5
10 a=INT(RND(1)*6)+1
20 INPUT "ｱﾅﾀﾉ ﾖｿｳ ﾊ ";b
22 IF FRAC(b)<>0 THEN 20 ELSE IF b>=1
   AND b<=6 THEN 30 ELSE 20
30 IF a=b THEN PRINT "ｱﾀﾘ !":x=x+1:GOTO
50
40 PRINT "ﾊｽﾞﾚ.."
50 NEXT i
60 PRINT "ｱﾀﾘﾉ ｶｲｽｳ=";x
Ok
renum
Ok
list
10 FOR i=1 TO 5
20 a=INT(RND(1)*6)+1
30 INPUT "ｱﾅﾀﾉ ﾖｿｳ ﾊ ";b
40 IF FRAC(b)<>0 THEN 30 ELSE IF b>=1
   AND b<=6 THEN 50 ELSE 30
50 IF a=b THEN PRINT "ｱﾀﾘ !":x=x+1:GOTO
70
60 PRINT "ﾊｽﾞﾚ.."
70 NEXT i
80 PRINT "ｱﾀﾘﾉ ｶｲｽｳ=";x
Ok
■
```

ほらね。プログラムが完成したら，この RENUM を入れてきれいに番号をそろえましょう。(☞33)

33

RENUMはリナンバーと言って、行番号をきれいにそろえてくれる命令です。RENUM [CR] だけ打つと、10番から10番おきにそろえてくれます。

RENUM 100,10,10
新行番号 旧行番号 増分

旧行番号で指定した行番号を、新行番号で始まる行番号に付け替えます。新しい行番号は、増分の一定間隔で増えます。GOTO、GOSUBなどの飛び先の行番号も同時に変化します。ただし、新行番号は旧行番号より大きい事。よろしいですか。

カズアテゲームを更に改変して、何回で"カズアテ"に成功するかのプログラムを…

さて，数あてゲームがどんどん長いプログラムになって来ちゃいましたが，大丈夫かな？　でも，ひとつずつ理解していけば，そんなに難しくはないみたい。

私にもついて行けるんですから……。でも，先生にも困ったものです。おもしろがって，どんどんプログラムを変えちゃうんだもの。

```
list
10 a=INT(RND(1)*100)+1
20 INPUT "ｱﾅﾀﾉ ﾖｿｳ";b
30 x=x+1
40 IF a=b THEN 80
50 IF a>b THEN PRINT "ﾁｲｻｽｷﾞﾙﾖ"
60 IF a<b THEN PRINT "ｵｵｷｽｷﾞﾙﾖ"
70 GOTO 20
80 PRINT x;"ｶｲﾃﾞ ﾀﾞｲｾｲｶｲ !!"
Ok
■
```

何回で"カズアテ"に成功するか？そのプログラムの構造は試行錯誤で的を絞って…

今度はどんなプログラムになっちゃったんでしょう。1行ずつ見てみましょうか。

まず，行番号10で１～100までの整数をでたらめに選ばせ，これがａとなります。

20番で，“アナタノヨソウ”と入力して，セミコロンのあとにｂを入れます。ゲームをする人にｂの値をたずねるわけです。

30番で，$x=x+1$ という前にやったあの数を数える時の式が出てきます。忘れちゃった人は，数あてゲームを10回やって統計をとった箇所（Ｐ60）に戻ってみてください。

40番からａとｂの大小比較です。40番では，ａ＝ｂならば予想があたったわけですから，80番のｘ；“カイデダイセイカイ!!”へ直接行けと命じています。50番，60番では，予想が小さすぎるか大きすぎる場合に，そう表示します。大きすぎるか，小さすぎた場合には，次に70番へきて，また20番の“アナタノヨソウ”；ｂへ戻り，くり返し聞いてくれます。

これがａ＝ｂになるまでくり返され，当たった時には，何回で当たったかが80番によりPRINT OUTされるんですヨ。以上でこのプログラムの説明は終わり！

さあ，はやくゲームをしてみましょうよ。

```
run
アナタノ ヨソウ? 50
オオキスギ゛ルヨ
アナタノ ヨソウ? 30
チイサスギ゛ルヨ
アナタノ ヨソウ? 40
オオキスギ゛ルヨ
アナタノ ヨソウ? 35
オオキスギ゛ルヨ
アナタノ ヨソウ? 34
オオキスギ゛ルヨ
アナタノ ヨソウ? 32
オオキスギ゛ルヨ
アナタノ ヨソウ? 31
 7 カイテ゛ タ゛イセイカイ !!
Ok
█
```

50番と60番はELSEでつなげて，１つの命令文にすることだって，もちろん可能ですよ！

```
50 if a>b then print "チイサスギ゛ルヨ"else prin
t "オオキスギ゛ルヨ"
```

これで60番は不要ですネ。

CLSは画面を消していく命令、a$の文字変数も使うとYes,Noの選択を…

ここで，画面を全部消してから始めなさいという命令を覚えておきましょうネ。**CLS**という命令です。これを5番に入れておくと，実行する時にきれいな画面が出てきます。更にゲームをおもしろくする為にもう2つ程命令文を加えてみましょうね。

```
list
5 CLS
10 a=INT(RND(1)*100)+1
20 INPUT "ｱﾅﾀﾉ ﾖｿｳ";b
30 x=x+1
40 IF a=b THEN 80
50 IF a>b THEN PRINT "ﾁｲｻｽｷﾞﾙﾖ"ELSE PRIN
T "ｵｵｷｽｷﾞﾙﾖ"
70 GOTO 20
80 PRINT x;"ｶｲﾃﾞ ﾀﾞｲｾｲｶｲ !!"
90 INPUT "ﾓｳｲﾁﾄﾞｱｿﾋﾞﾏｽｶ (y or n)";a$
100 IF a$="y" THEN 5
110 PRINT "ﾊﾞｲ ﾊﾞｲ ..."
Ok
■
```

(☞34)
(☞35)

ここで，90，100，110が新しく加わりました。90番でa$が出てきましたネ。$は覚えてますか。文字変数でしたよね。a$が“y”ならば，5番に戻ってもう1度画面を消しなさいという命令を，100で与えているわけです。

```
ｱﾅﾀﾉ ﾖｿｳ? 50
ｵｵｷｽｷﾞﾙﾖ
ｱﾅﾀﾉ ﾖｿｳ? 30
ﾁｲｻｽｷﾞﾙﾖ
ｱﾅﾀﾉ ﾖｿｳ? 40
ﾁｲｻｽｷﾞﾙﾖ
ｱﾅﾀﾉ ﾖｿｳ? 45
ﾁｲｻｽｷﾞﾙﾖ
ｱﾅﾀﾉ ﾖｿｳ? 48
ｵｵｷｽｷﾞﾙﾖ
ｱﾅﾀﾉ ﾖｿｳ? 47
 6 ｶｲﾃﾞ ﾀﾞｲｾｲｶｲ !!
ﾓｳｲﾁﾄﾞｱｿﾋﾞﾏｽｶ (y or n)? n
ﾊﾞｲ ﾊﾞｲ ...
Ok
■
```

34

IF文の後のGOTOは省略できます。

```
IF A=B THEN GOTO 80
IF A=B THEN 80
```

どちらも同様の動きをします。

35

"ﾓｳｲﾁﾄﾞ ｱｿﾋﾞﾏｽｶ(YorN)"

この時YとNは大文字か小文字かをよく確かめて入力してください。大文字だけ判断しているプログラムは、小文字の入力は受け付けません。もしどちらも判定するためには、

```
IF A$="Y" OR A$="y"
THEN 5
```

と書きかえてください。

今度は家族構成を記憶させるプログラム、PRINT文とIF文が中心です

さて，先生が今度は，私の家族構成をコンピュータに記憶させておこうと言い出しました。どんな風になっちゃうのかしら？

```
new
Ok
■
```

```
10 CLS
20 PRINT "** ｼﾅｶﾞﾜ ｹ ﾉ ｶｿﾞｸ **"
30 PRINT "ｼﾅｶﾞﾜ ｹｲｿﾞｳ =1"
40 PRINT "ｼﾅｶﾞﾜ ｽﾐｴ    =2"
50 PRINT "ｼﾅｶﾞﾜ ｷﾇ     =3"
60 PRINT "ｼﾅｶﾞﾜ ﾕﾘ     =4"
70 PRINT "ｼﾅｶﾞﾜ ﾅﾂ     =5"
80 INPUT "ﾀﾞﾚ ｦ ﾐﾏｽｶ ";a
90 IF a=1 THEN 200
100 IF a=2 THEN 300
110 IF a=3 THEN 400
120 IF a=4 THEN 500
130 IF a=5 THEN 600
200 CLS
210 PRINT "ｼﾅｶﾞﾜ ｹｲｿﾞｳ"
220 PRINT "ﾁﾁ      56ｻｲ"
230 INPUT "ﾖﾛｼｲﾃﾞｽｶ (y or n)";a$
240 IF a$="y"THEN 10
250 GOTO 200
300 CLS
310 PRINT "ｼﾅｶﾞﾜ ｽﾐｴ"
320 PRINT "ﾊﾊ      47ｻｲ"
330 INPUT "ﾖﾛｼｲﾃﾞｽｶ (y or n)";a$
340 IF a$="y"THEN 10
350 GOTO 300
```

(☞36)

```
400 CLS
410 PRINT "ｼﾅｶﾞﾜ ｷﾇ"
420 PRINT "ﾁｮｳｼﾞｮ 24ｻｲ"
430 INPUT "ﾖﾛｼｲﾃﾞｽｶ (y or n)";a$
440 IF a$="y"THEN 10
450 GOTO 400
500 CLS
510 PRINT "ｼﾅｶﾞﾜ ｺﾘ"
520 PRINT "ｼﾞｼﾞｮ  20ｻｲ"
530 INPUT "ﾖﾛｼｲﾃﾞｽｶ (y or n)";a$
540 IF a$="y"THEN 10
550 GOTO 500
600 CLS
610 PRINT "ｼﾅｶﾞﾜ ﾅﾂ"
620 PRINT "ｻﾝｼﾞｮ  18ｻｲ"
630 INPUT "ﾖﾛｼｲﾃﾞｽｶ (y or n)";a$
640 IF a$="y"THEN 10
650 GOTO 600
```

36

このようなプログラムのことを、ファイル検索プログラムと言って、色々な応用が考えられます。電話帳、レコードの整理等、楽しいですぞ。

ON〜GOTO（次に行く番号を指定する）文を使ってIF文を1つにまとめたら…

ずいぶん長ぃプログラムですねぇ。これなんとか短かくならないものでしょうか？　例えば，90〜130までの1F文を，どうにか一つにまとめられないものかしら？　何度も同じ事を聞くのは大変よね！　そこで次の命令に注目！

```
90 on a goto 200,300,400,500,600
```

```
10 CLS
20 PRINT "** ｼﾅｶﾞﾜ ｹ ﾉ ｶｿﾞｸ **"
30 PRINT "ｼﾅｶﾞﾜ ｹｲｿﾞｳ =1"
40 PRINT "ｼﾅｶﾞﾜ ｽﾐｴ   =2"
50 PRINT "ｼﾅｶﾞﾜ ｷﾇ    =3"
60 PRINT "ｼﾅｶﾞﾜ ﾕﾘ    =4"
70 PRINT "ｼﾅｶﾞﾜ ﾅﾂ    =5"
80 INPUT "ﾀﾞﾚ ｦ ﾐﾏｽｶ ";a
90 ON a GOTO 200,300,400,500,600
```

便利でしょ。これが ON…GOTO です。これは，aに入る変数によりどこへ行くかが決まってくるんですって。でも，まだこのプログラム長いわよねぇ。同じ事くり返して聞いてるし，なんとかなりそうネ。先生がずいぶん短かいプログラムに書きかえてくれたんだけど，すごく難しそうなのよね。ちょっと見てください。(☞37) (☞38)

37

ON〜GOTO文

```
ON A GOTO 100,200,300
```

Aの値が1の時は100番へ、2の時は200番へとジャンプして、それ以降の行を実行します。Aの値に対応する行番号がなければ、ON文の次の実行を行います。なおAの値が整数でなければ、小数第一位で四捨五入された値になります。もちろんこのAは変数ですので、どんな変数でもかまいませんよ。

```
ON YURI GOTO 100,200
,300
```

38

いよいよ本格的データ・ベースプログラムに挑戦していただきます。D-IM、READ〜DATA、この命令を使いこなせたら、もう一人前なんですぞ。しっかり勉強してくださいね。

DIM(ディメンジョン——次元)は配列変数と呼び、文字変数の範囲を指します

```
10  DIM s$(5),z$(5),t(5)
20  CLS:PRINT "** ｼﾅｶﾞﾜ ｹ ﾉ ｶｿﾞｸ **"
30  s$(1)="ｼﾅｶﾞﾜ ｹｲｿﾞｳ":z$(1)="ﾁﾁ":t(1)=5
6
40  s$(2)="ｼﾅｶﾞﾜ ｽﾐｴ  ":z$(2)="ﾊﾊ":t(2)=4
7
50  s$(3)="ｼﾅｶﾞﾜ ｷﾇ   ":z$(3)="ﾁｮｳｼﾞｮ":t(
3)=24
60  s$(4)="ｼﾅｶﾞﾜ ﾕﾘ   ":z$(4)="ｼﾞｼﾞｮ":t(4
)=20
70  s$(5)="ｼﾅｶﾞﾜ ﾅﾂ   ":z$(5)="ｻﾝｼﾞｮ":t(5
)=18
75  FOR i=1 TO 5
76  PRINT s$(i);"=";i
77  NEXT i
80  INPUT "ﾀﾞﾚ ｦ ﾐﾏｽｶ ";a
200  CLS
210  PRINT s$(a)
220  PRINT z$(a),t(a);"ｻｲ"
230  INPUT "ﾖﾛｼｲﾃﾞｽｶ (y or n)";a$
240  IF a$="y"THEN 20
250  GOTO 200
```

```
** ｼﾅｶﾞﾜ ｹ ﾉ ｶｿﾞｸ **
ｼﾅｶﾞﾜ ｹｲｿﾞｳ= 1
ｼﾅｶﾞﾜ ｽﾐｴ  = 2
ｼﾅｶﾞﾜ ｷﾇ   = 3
ｼﾅｶﾞﾜ ﾕﾘ   = 4
ｼﾅｶﾞﾜ ﾅﾂ   = 5
ﾀﾞﾚ ｦ ﾐﾏｽｶ ? ■
```

```
ｼﾅｶﾞﾜ ﾕﾘ
ｼﾞｼﾞｮ         20 ｻｲ
ﾖﾛｼｲﾃﾞｽｶ (y or n)? ■
```

```
ｼﾅｶﾞﾜ ｹｲｿﾞｳ
ﾁﾁ            56 ｻｲ
ﾖﾛｼｲﾃﾞｽｶ (y or n)? ■
```

この DIM はディメンジョン（次元）配列変数です。最初から知らない命令が出てきて，とまどいませんでした？　辞書をひいてみると，次元という訳語が出ていました。(☞39)

これを使えば，文字変数の数の範囲が指定できるんです。このプログラムはちょっと難しいので，配列変数のみをわかりやすく示すようなプログラムを考えてみました。

39

DIM ディメンジョン(配列変数)

DIM A(10)　　DIM A$(10)

配列名 添字　文字配列名 添字

配列変数の名前と、その添字の上限を設定します。1つのDIMで、複数の配列変数を定義することができます。DIMの実行後、数値型配列には0が入り、文字型配列は、ヌルストリングス("")になります。

配列変数の添字の上限を越えた指定を実行すると、

DIM A(10) なのに、

A(11)=1　などとすると、

```
Subscript out of ran
ge
Ok
▒
```

のエラーメッセージが出てきますのでご注意下さい。

配列は変数を入れておく箱なのです。この箱を何個使うかを、初めに宣言しておくのがDIM文なのです。

1	2	3	4	5	6	7	8	9	10

Aは10個ですよ

この配列の中には2次元、3次元など、メモリーのあるかぎり指定することができます。たとえば、

A(10,10)　**2次元配列**

	1	2	3	4	5	6	7	8	9	10
1										
2							◆			
3										
4				♥						
5										
6										
7						♣				
8										
9										
10										

この様に10×10の箱を用意するのです。

◆ = A(7,2)

♥ = A(4,4)

♣ = A(6,7)

と、それぞれのデーターをしまっておく整理箱なのです。

また、3次元配列はルービックキューブを考えて下さい。

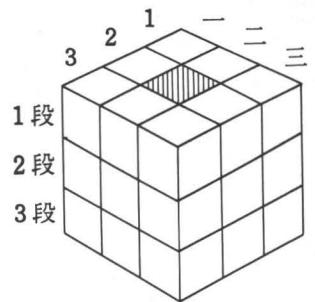
DIM文で指定した配列数s$(n)をFOR~NEXT文でデータ配列指定ができます

```
list
10  DIM s$(5)
20  s$(1)="シナカ゛ワ ケイゾ゛ウ"
30  s$(2)="シナカ゛ワ スミエ"
40  s$(3)="シナカ゛ワ キヌ"
50  s$(4)="シナカ゛ワ ユリ"
60  s$(5)="シナカ゛ワ ナツ"
70  FOR i=1 TO 5
80  PRINT s$(i)
90  NEXT i
Ok
run
シナカ゛ワ ケイゾ゛ウ
シナカ゛ワ スミエ
シナカ゛ワ キヌ
シナカ゛ワ ユリ
シナカ゛ワ ナツ
Ok
■
```

このように，はじめにデイメンジョンで配列数をあげ，その範囲内でFOR−NEXTによって配列を指定できるわけです。と言ってもよく分からないでしょ。順を追って説明すると，10番のDIM s$（5）でs$の範囲は5であること。つまりs$のグループには5つのデータが入っていることが示されています。DIMとは順序づけを行った値のリストで，私達はその中から好きなデータを取り出すことが可能なんです。

（ ）内の数は**次元**と呼び，これが1つなら1次元，2つなら2次元と言います。ちなみにこのプログラムは1次元ストリング配列です。(DIM s（5）ならば1次元数値配列，DIM s$（5）なら1次元ストリング配列になります)

1次元ストリング配列であるこのプログラムの中から，部分的にデータを取出してみましょう。

```
70 for i=3 to 4
run
シナカ゛ワ キヌ
シナカ゛ワ ユリ
Ok
■
```

これは、A（3，3，3）と3次元配列になりますね。上面の真中は、1段目の2列目の2ですからA(1，2，2)という箱になるわけです。このように、DIMは変数の収納箱のことなのです。

ほらね。こういう事もできるわけですよ。配列されているデータの中からだったら，どんな形でもデータは取り出せます。プログラムの中で20番から60番がデータの箇所です。

DIM文で配列以外の数字を指定すると、当然のことながらERRORが出ます

```
list
10  DIM  s$(5)
20  s$(1)="ｼﾅｶﾞﾜ ｹｲｿﾞｳ"
30  s$(2)="ｼﾅｶﾞﾜ ｽﾐｴ"
40  s$(3)="ｼﾅｶﾞﾜ ｷﾇ"
50  s$(4)="ｼﾅｶﾞﾜ ﾕﾘ"
60  s$(5)="ｼﾅｶﾞﾜ ﾅﾂ"
70  FOR  i=1  TO  6
80  PRINT  s$(i)
90  NEXT  i
Ok
run
ｼﾅｶﾞﾜ ｹｲｿﾞｳ
ｼﾅｶﾞﾜ ｽﾐｴ
ｼﾅｶﾞﾜ ｷﾇ
ｼﾅｶﾞﾜ ﾕﾘ
ｼﾅｶﾞﾜ ﾅﾂ
Subscript out of range in 80
Ok
■
```

配列外の数まで要求すると，上のような表示が出て誤りである事を教えてくれます。10できめる配列変数と70で指定する数の範囲がくい違っていてはいけないわけですよね。

複数の配列変数(S.Z.T)を使えば、データを各ブロックに分類定義できます

```
75 if i<1 or i>5 then 20
```

最後にこのようにリミットをもうけておけば，さっきのようなエラーのメッセージは出ませんよね。

配列変数のポイントはわかりましたね。それでは，さっきの

最初のプログラムに戻ってみましょう。

このプログラムの10番には，３つの配列変数が使われています。違う文字であれば，このように幾つものブロックとして，データを分類して定義することが可能です。

ｓは…SHINAGAWA ということでＳをつかいます。

ｚは…つづき柄

ｔは…年令

（㊟年令の場合は数字変数，このDIM ｔ（ ）は一次元数値配列でいいので＄はつけませんでした）

という事で入れてみました。

今度は**READ-DATA**という命令を利用して，もっとプログラムを簡単にしてみましょう。READ と DATA は必ずペアで使って下さいネ！（☞40-A）

```
list
10 DIM s$(5),z$(5),t(5)
20 FOR i=1 TO 5
30 READ s$(i),z$(i),t(i)
40 NEXT i
50 s$="ｼﾅｶﾞﾜ "
60 CLS:PRINT "** ｼﾅｶﾞﾜｹ ﾉ ｶｿﾞｸ **"
70 FOR i=1 TO 5
80 PRINT s$;s$(i);"=";i
90 NEXT i
100 INPUT "ﾀﾞﾚｦ ﾐﾏｽｶ";a
110 IF a<1 OR a>5 THEN 60
120 CLS
130 PRINT s$;s$(a)
140 PRINT z$(a);t(a);"ｻｲ"
150 INPUT "ﾖﾛｼｲﾃﾞｽｶ (y or n)";a$
160 IF a#="y"THEN 60
170 GOTO 120
1000 data "ｹｲｿﾞｳ","ﾁﾁ",56,"ｽﾐｴ","ﾊﾊ",47,
"ｷｸ","ﾁﾖｳｼﾞﾖ",24,"ﾕﾘ","ｼﾞｼﾞﾖ",20,"ﾅﾂ","ｻ
ﾝｼﾞﾖ",18
Ok
■
```

DATA はどの位置でもかまいません。コンピュータはどこにあっても READ の命令があれば DATA を探すんです！READ～DATAを使ったおかげで命令が整理されて大分きれいになったでしょ。

40—A

READ DATA

DATA文で用意されたデーターを読み込んで、変数に割り当てます。READ文はいつもDATA文と対にして使い、DATAの定数データーは、READ文の変数と１対１に対応します。また、それらは同じ型でなければいけません。１つのREAD文で２つ以上のDATAからデータを読み込んだり、いくつかのREAD文で、１つのDATA文を共用して読み込むことができます。

いずれの場合も、DATA文は行番号の若い順にデータの並びの先頭から読み込まれ、DATA文の定数データの数が、READ文の変数の数より多いときには、次のREAD文により引き続き読み込まれ、READ文がなければ、残りの定数データは無視されます。

データの数が不足しているときは、

`Out of data`

のエラーメッセージが表示されます。

READ~DATA文の内容を実際にナンバーどおりに呼び出してみました

じゃあ，もう少し READ~DATA のお勉強をしましょうネ。

```
list
10 DIM a$(5)
20 FOR i=1 TO 5
30 READ a$(i)
40 NEXT i
50 INPUT a
60 PRINT a$(a)
70 GOTO 50
100 DATA "ｼﾅｶﾞﾜ","ﾖｺﾊﾏ","ｺｳﾍﾞ","ﾀﾏ","ｻｯﾎﾟﾛ"
Ok
■
```

私の名前に合わせて，DATA は車のナンバーにしてみました。いいアイディアでしょ！(☞40-B)

RUN してみましょう。

```
run
? 1
ｼﾅｶﾞﾜ
? 5
ｻｯﾎﾟﾛ
? 3
ｺｳﾍﾞ
? ■
```

DATA は並べた順に 1，2，3…となります。この命令も，上手に活用すれば大変便利なものなんですョ。(☞41)

もしデータの数が20番で指定されたものに足りなければ，RUN すると……，

```
Out of data in 30
Ok
■
```

というような表示が現れます。

40—B

文字列をDATAとして用意する場合、ダブルクォーテーション(〃 〃)で囲まなくても使用することができます。

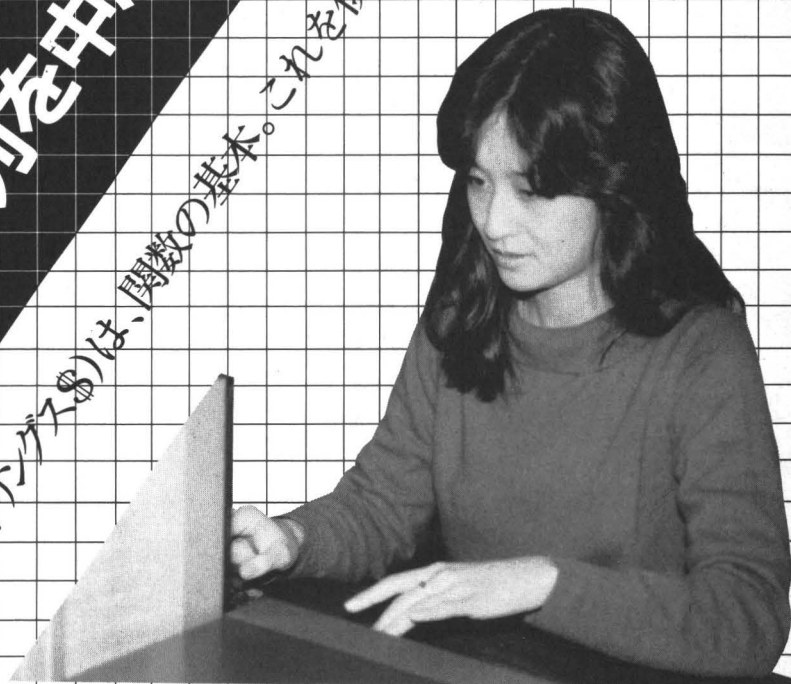
DATA ｼﾅｶﾞﾜ,ﾖｺﾊﾏ,ｺｳﾍﾞ,ﾀﾏ,ｻｯﾎﾟﾛ

41

RESTORE (リストア)

DATAを一度読み、再度読み込む必要が出て来た時は、この命令でデータの初期化を行います。

DATA文は、一度読まれたら２度と読むことは不可能です。しかし、このRESTOREを実行することにより、再度読み込ませることが出来るのです。この本の最後の応用プログラム〝世界の国々〟を参考にしてみてください。

NOTE4
文字列を中心とした特殊関数
文字列(ストリングス$)は、関数の基本。これを使いこなしてプログラミング力をアップ

STRING$は“”で囲まれた文字の集まり。通常変数との相違点に気をつけてネ!

だんだんおもしろくなってきましたか？　それとも，わからなくなってきちゃった？　でも，ここであきらめるのはまだ早い!　さあがんばりましょう。

次に勉強するのはストリング処理関数です。ストリングについては，もう触れていますね。

一言で言ってしまえば「“ ”」で囲まれた文字の集まりなんですって。ストリング変数だって，もう知ってるでしょう。A＄とかB＄とか＄のマークはすでにたくさん出てきています。

でも，なんだかごちゃごちゃして来たなぁ，と思う方もいるでしょうね。

例えば，セミコロンとコンマだけとってみても，ストリング変数と，通常の変数では違うんですヨ。

◦ **ストリング変数**

「；」は，文字と文字の間に空白をあけません。

「，」は10文字分の一定幅の中に文字を書き出します。

◦ **通常変数**

「；」は，数値と数値の間に空白を1字分あけます。

「，」は10文字分の一定幅の中に数値を書き出します。

こんな風に頭の中で，整理しておけばわかりやすいですね。

STRINGの合成、これはSTRINGのたし算で、Yuri Shinagawaも簡単ですヨ!

さあ，ではいよいよストリングの加工のお話だそうですよ!

まずストリングの合成について，これはいくつかのストリングをストリングのたし算を使って，新しいストリングにしてしまう方法です。ようするにたし算なんだから，あまり難しく考えなくていいんじゃないかしら。

```
list
10 a$="Yuri"
20 b$=" Shinagawa"
30 c$=a$+b$
40 PRINT c$
Ok
run
Yuri Shinagawa
Ok
■
```

こんなわけです。もし Yuri Shinagawa の間に1字分空白を置きたければ“ Shinagawa”とすればいいんですよ。

A$+B$+C$をPRINTにすれば、STRING3つの合成でYuri and Kikuも…

応用で，こんな事もできます。

```
list
10 a$="Yuri"
20 b$=" and "
30 c$="Kiku"
40 PRINT a$+b$+c$
Ok
run
Yuri and Kiku
Ok
■
```

ほらね!ちゃんとスペースができたでしょ。ストリング変数も通常の変数も，考え方の上で大きな差はありません。

READ~DATA文はINPUT A$をPRINT A$で出すのと同じに考えて…

READ～DATA 文，INPUT 文は全く同じように考えられるそうですよ。

```
10 READ a$
20 PRINT a$
30 DATA MEDIA
run
MEDIA
Ok
■
```

ここで DATA 記号でストリングを与える時は，クォーテーションマークがいらないという事は憶えておいてください（30番の部分)。あとは特に新しい事はないでしょ。

```
10 INPUT a$
20 INPUT a
30 INPUT b
40 PRINT a$;(a+b)/2
run
? Heikin
? 2
? 10
Heikin 6
Ok
■

10 input a$,a,b
20 print a$;(a+b)/2
run
? Heikin,2,10
Heikin 6
Ok
■
```

LEFT$(A$, B)はSTRINGの文字を左端からB番目まで抜き出すコマンド

今度は，ストリングの頭（左端）から数えて，何番目かの文字までをとり出して，新しいストリングが作れるというお話。

この命令は，LEFT$（A$, B）という形でいいんですって。これで，A$の左から数えてB番目までをストリングとするという意味なんです。数えはじめるのはダブルクォーテーションの次，つまり文字の頭からですよ。(☞42)

```
10 a$="Yuri Shinagawa"
20 PRINT LEFT$(a$,4)
run
Yuri
Ok
█
```

42

LEFT$　レフトダラー

LEFT$("ABCD",2)

"ABCD"の文字列の左から2ケ取り出しなさい、という命令です。ですからこれはABということですね。変数だけじゃなく、このように直接ダブルクォーテーションで書き込んでもOKですよ。

LEFT$(A$,B)ではBがSTRINGの文字数より多いと、STRINGがまるまる出て

これは問題ないわよねぇ。ここで，ちょっと意地悪な事を考えちゃいました。もしストリングの文字数より大きなものを指定しちゃったらどうなるのかしら…。コンピュータも悩むだろうなあ？

```
10 a$="Yuri Shinagawa"
20 b$=LEFT$(a$,20)
30 PRINT b$
run
Yuri Shinagawa
Ok
■
```

LEFT$(A$,B)では、Bのかたちいかんを問わず、0≦B≦255を充さなければ…

なるほど，Bがストリングの文字数より大きい数のときは，ストリングをそのまま出力してくれるんですね。
じゃあBは100でも1,000でも何でもいいのかしら。

あれれ，先生が首を横に振っています。

LEFT$ (A$, B) では，Bが数値変数，数値定数，計算式のいずれを問わず0≦B≦255を充さなければいけないそうですよ。B=0のときは，RUNしても何も表示してくれません。また，Bは計算式でも良いわけだからLEFT$ (A$, A+B) とか，LEFT$ (A$, A/B) なんてことも，もちろん可能。

```
10 a$="246810"
20 a=2
30 b=1
40 PRINT LEFT$(a$,a+b)
run
246
Ok
■
```

LEFTがあるのだから、RIGHTもMIDもあってかまわない。MIDは途中から何字出すかの…

さっきから考えてたんだけど，LEFT があるなら，RIGHT もあっていいはずよねえ。先生に聞いてみました。

そうしたら，RIGHT もあるし，MID もあるんですって。MID っていうのは……？ ああそうか／ きっと途中から始める事ね。英語の MID を引いてみると……，フムフム…… “中央の”，“中部の”，“中間の” ですって。

MID$ (A$, B, C) は，B番目からC個の数を取れという命令だそうです。

```
10 a$="ONAKASUITANA"
20 PRINT MID$(a$,4,5)
run
KASUI
Ok
■
```

とこんな具合に，B番目からC個とって新しいストリングを作ってくれます。今度もB，Cは，0≦B，C≦255 を満たしていればいいそうですよ。

```
10 a$="ONAKASUITANA"
20 b=2
30 c=4
40 PRINT MID$(a$,2,b+c)
run
NAKASU
Ok
■
```

このように，B，Cは計算式でもなんでもよいなどの点は，LEFT$ (A$, B) に同じです。(☞43)

43

RIGHT$　ライトダラー

```
RIGHT$("ABCD",3)
```

“ABCD”の右から3個目から最後まで取り出しなさい、ということですね。

BCD

MID$　ミッドダラー

```
MID$("ABCDE",2,2)
```

“ABCDE”の左から2個目から2つだけ取り出しなさい、ということですね。

BC

このように、文字を自由に組立てる便利な命令ですよ。

RIGHT$(A$,B)は右端からB字分出すのだけど、表示は左から出てくるの

今度は **RIGHT$**。もうわかりますね。でも早合点は失敗のもと。RIGHT$（A$, B）は，A$の最終文字から左に数えてB文字分取り出す命令なんですが，B字分数えて表示はちゃんと左からしてくれます。

```
10 a$="Yuri Shinagawa"
20 PRINT RIGHT$(a$,5)
run
agawa
Ok
■
```

数える時は← 5 4 3 2 1
表示する時は 1 2 3 4 5 →

よく **awaga** となるんじゃないか？なんて考える人がいますが，5文字数えてからちゃんと左から右に出してくれます。コンピュータはおりこうなんですね。

あとは，全部LEFT, MIDと同じ考え方。さあ，これでまた3つ命令がふえました。

SPACE$(X)はPRINT文中にスペース間隔をX個文あけてくれるものなのです

この命令の応用のような形で **SPACE$**（X）も覚えましょう。これは，PRINT文中にSPACEをX個分あけてくれるものなんだそうですよ。（☞44）

たとえば，こんなプログラムになるのかな。

```
10 PRINT "Yuri";SPACE$(5);"Shinagawa"
```

RUNしてみると

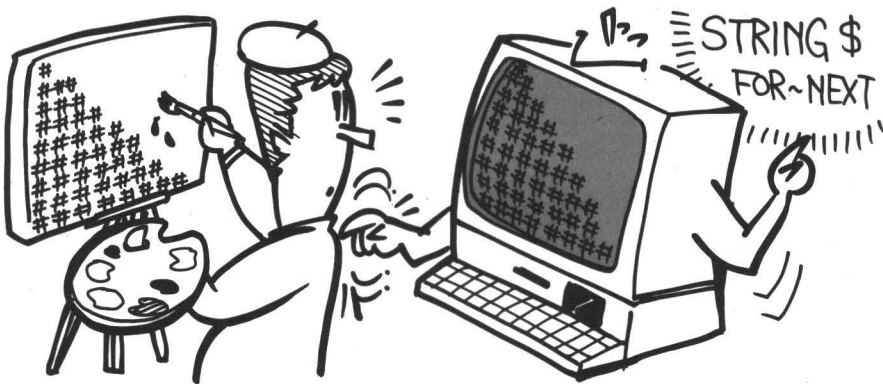
となりました。

44

SPACE$　スペースダラー

これはスペースつまり空白の個数を指定する命令です。

SPACE$ (10) とすると、空白を10個という意味です。このスペースの数は0から255個まで指定することができます。表の頭をそろえたりするのに便利ですね。

```
#
##
###
####
#####
```

次にこんな模様（#）を OUTPUT するために **STRING$**(B, A$）なる命令を使ってみようと考えました。これは，どういうものなんでしょうか。(☞45)

```
list
10 PRINT STRING$(7,"#")
Ok
run
#######
Ok
■
```

なるほど。

これはストリング変数，あるいは定数A$をB個分つなぐものらしいんです。

これで，こんな模様を作りたいんだけれど，先生はどんなプログラムを作ったのかしら。先生はいじわるだから教えてくれないと言ってますよ。

いいわ！ それなら自分で考えるから……。

45
STRING$ ストリングダラー

```
PRINT STRING$(3,"YURI")
```

この様に指定すると、

```
YURIYURIYURI
```

と3回書きなさい、ということなのです。

この指定は、0から255個まで指定できます。任意の文字列を任意の数だけ用意しなさい、という意味ですね。

```
A$="###############"
```

なんてたくさん書かなくても、

```
A$=STRING$(15,"#")
```

でOKなのです。同じ文字をたくさん用意したい時に便利ですね。

このSTRING$の文字列は複数個の文字を扱えますが、HuBASIC以外のBASICでは、この文字列は1文字にかぎられております。

FOR～NEXTを使ってA$＝"#"としたら、STRING$(I, A$)で#が#,##,###と増えて…

#, ##, ###, こんな風にひとつずつ記号がふえてるんだから, プログラムの中にはきっと FOR～NEXT 文が使われているはずよね。

```
list
10  a$="#"
20  FOR i=1 TO 5
30  PRINT STRING$(i,a$)
40  NEXT i
Ok
■
```

ここではFOR～NEXT 文Iが1から始まって5までふえつづけ, 30番のA＄には10番の#が入るから, どうかなあ？　たぶんいいと思うんだけどなあ……。

```
run
#
##
###
####
#####
Ok
■
```

やった！　ほんとにFOR～NEXTって便利。(☞46)

46

なるほど、良くできました。だが、このプログラムは、MID$を使った方が流れは良いですな。

```
10 A$=STRING$(5,"+")
20 FOR I=1 TO 5
30 PRINT MID$(A$,1,I)
40 NEXT I
```

LEN(A$)で、A$の文字数を教えてくれるのです。LEN(A$+B$)もできます

まだまだ便利な命令はあるそうですよ。LEN（X$）はストリング変数の構成文字数を数えてくれるんですって。この場合，空白も一字と数えるみたいです。

```
10 a$="Yuri"
20 b$="Shinagawa"
30 PRINT LEN(a$+b$)
run
 13
Ok
■
```

FOR～NEXTでI=1 to LEN(A$)としておいて、PRINT"＊"で出せば＊が文字数だけ出ます

それでは，LEN を利用して，ちょっとおもしろいプログラムを考えてみましょう。

```
10 a$="ATSUI"
20 PRINT a$
30 FOR i=1 TO LEN(a$)
40 PRINT "*";  ―― 横に並らべる
50 NEXT i
```

さあどんな結果になると思います。

```
run
ATSUI
*****
Ok
■
```

ねっ，おもしろいでしょ。これはまず行番号20で ATSUI と書かせて，30番で文字数を数えさせ，FOR～NEXTで1からA$の文字数になるまで処理させて，つづけて＊を表示させたんです。（☞47）

47

この文字変数の中に何個の文字が入っているのかな。そんな時には、

LEN(X$)

を使いましょう。

```
X$="12345 67890"
PRINT LEN(X$)
 11
Ok
■
```

そうです。スペースも文字のひとつなのですよ。

STRING命令に/や¥を入れて////や¥¥¥¥を出したり、かけ算やたし算を組み合わせて…

STRING＄ (N, X＄) は，グラフや図表を作る場合に役立つようです。

とっても楽しくて，たくさんプログラムを作ったので挙げておきます。

```
10 print string$(10,"/")
run
//////////             ―――― /印を10個つないでくれます
Ok
10 a$="¥"
20 print string$(10,a$)
run
¥¥¥¥¥¥¥¥¥¥             ¥印を10個つなぐ
Ok
■
```

ストリング定数を使う時は/のプログラムのようにして，クォーテーションマーク「“ ”」でかこむか，¥のようにストリング変数を使うかの２通りです。(☞48)

```
10 print string$(0,"*")
run
                       0の時は何も出力しない
Ok
10 a$="-"
20 x=3
30 print string$(x*2,a$)
run
------                 A$の“ー”を３＊２個、つまり
                       6個つなげて出力
Ok
10 a$="Tom Betty"      1文字として数える
20 print len(a$+" Jack")
30 print a$;" Jack"
run
 14                    A$とストリングを足して
Tom Betty Jack         構成文字数を数える
Ok
■
```

48

文字の左から、右から、真中から、また文字は何個かな。同じ文字を続けたい時など、そう、色々の応用ができるのです。

HuBASICは文字の扱いに関しては、最高のBASICなのです。これからがもっともっと面白くなるのですぞ。

NOTE5
関数の征服にチャレンジ!!
いろいろな関数を、この際徹底的にものにしてしまおう

今日は秋葉原に見学に行って来ました。家電品を買う時に，父や母と一緒に行った事があるぐらいで，ほとんど知らないんですが，どんなお店があるんでしょうね。

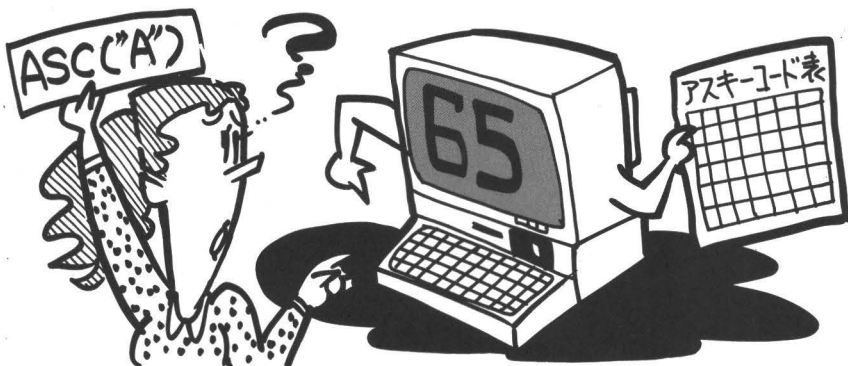
今日，初めて行ってみたのは九十九電機の7号店でした。女の子だけのマイコンショップと聞いていたので，どんな所なのか，すごく興味があったんですよね。店内に入ると，赤いベスト・スーツで統一した女性達が，お客さんと一緒になって，コンピュータを操作しているのが目につきます。

女子高校生の書いたマイコンの本なんかもあったし，女性も結構がんばってますよ！

でもやっぱりマイコン・ブームなんですねえ。小中学生から大人まで，店内はかなり混み合っていました。みんなが興味を持ってるんだなあとあらためて実感！

秋葉原中に，マイコン shop が続出していて，お店の側でも，お客さんの needs に合わせようと懸命なのがよくわかります。

テレビゲームがスゴイ人気でゲームセンターが全盛だった頃があったけれど，今度はそれを自分でやっちゃおうという事なんでしょうね。

与えられたゲームをやっていくうちに，自分にもゲームが作れるんじゃないかと思うようになったのかしら。とにかくマイコン少年が多いのにはびっくり。さあ，彼らに負けないように，勉強をつづけましょう。

ASCIIコードって、10進数、記号、文字などをコンピュータに理解させるコードなの

ストリング処理関数は，まだまだあります。次に登場するのは **ASC**（ストリングス）。

これを表示するのに用いられるのが **ASCII コード**だそうです。何のことやら，よくわかりませんね。ASC（ストリングス）によって，ストリングの最初の符号に対する ASCII コードを表示するらしいんですが，そもそも ASCII（アスキー）コードって何でしょうか？

これは，American Standard Code For Information Interchange の略で，タイプなどから入力される10進数，記号，アルファベットなどを，コンピュータが理解できる数値コードに変換するものだそうです。

コード表は，次のページの通りです。

こんな事まで，すべて記憶していて，さっと変換してくれるんですから，たいしたものだと思いません？（☞49）

49

アスキーコード

X１のキーボードから画面に向ってプログラミング。すると画面には当然のごとく文字が表われますね。でも、これらの文字の形をそのままおぼえさせるとたいへんな量のメモリーが必要になるのです。もちろん処理速度も低下してしまいます。そこでＡＳＣIIコードとよばれる箱の中に、１文字ずつの符号でコンピュータに覚えさせているのです。このＡＳＣIIコードは、ほとんどのコンピュータに共通のコードで、世界中で使われております。

このアスキーコードの集まりが、プログラムになるのですよ。

この箱の番号を見てみましょうか。コード表の横の番号と縦の番号（0123456789ABCDEF）は16進数ですね。アスキーコードは横から先に読んで、例えばAのアスキーコードなら、16進数では41となります。これを10進数に直すと、

→0…F　10　11…19　1A…1F　20…29　2A…2F　30…3F　40　41

→0…15　16　17…25　26…31　32…41　42…47　48…63　64　65

このように65がアスキーコードとなるんです。

アスキーコード表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	@	P		0	@	P	`	p	▁	─		ー	タ	ミ	●	▒
1	A	Q	!	1	A	Q	a	q	▂	┴	。	ア	チ	ム	○	土
2	B	R	"	2	B	R	b	r	▃	┬	「	イ	ツ	メ	♠	金
3	C	S	#	3	C	S	c	s	▄	┤	」	ウ	テ	モ	♥	木
4	D	T	$	4	D	T	d	t	▅	├	、	エ	ト	ヤ	♦	水
5	E	U	%	5	E	U	e	u	▆	┼	・	オ	ナ	ユ	♣	火
6	F	V	&	6	F	V	f	v	▇	│	ヲ	カ	ニ	ヨ	◢	月
7	G	W	'	7	G	W	g	w	█	─	ァ	キ	ヌ	ラ	◣	日
8	H	X	(	8	H	X	h	x	▉	┐	ィ	ク	ネ	リ	╳	時
9	I	Y	)	9	I	Y	i	y	▊	└	ゥ	ケ	ノ	ル	■	分
A	J	Z	*	:	J	Z	j	z	▋	┌	ェ	コ	ハ	レ	▘	秒
B	K	ε	+	;	K	[	k	{	▌	╮	ォ	サ	ヒ	ロ	▚	年
C	L	→	,	<	L	¥	l	|	▍	╰	ャ	シ	フ	ワ	▞	円
D	M	←	-	=	M	]	m	}	▎	╯	ュ	ス	ヘ	ン	▝	人
E	N	↑	.	>	N	^	n	‾	▏	╭	ョ	セ	ホ	゛	▚	生
F	O	↓	/	?	O	_	o	π	╱	╲	ッ	ソ	マ	゜	□	〒

A$="Yuri"でASC(A$)をPRINTにしても、頭文字Yのコード89が出てきて…

ASC（X$）を使ったいくつかのプログラムを書いてみましょう。（☞50）

```
10 print asc("A")
20 b$="B"
30 print asc(b$)     ―― b$のBをアスキーコードに変換
run
 65 ―――― アルファベッドAのコード番号
 66 ―――― アルファベットBのコード番号
Ok
■
```

このプログラムでもわかる通り，ストリングは「" "」でかこまれていなければ実行されませんから注意してくださいね。

```
10 a$="Yuri"
20 print asc(a$)
run
 89
Ok
■
```

この場合，ストリング変数の中身の最初の一文字しか表示してくれません。89とはYに対するアスキーコードなんですね。

50

ASC("A")

この様に命令すると

```
PRINT ASC("A")[CR]
 65
Ok
▒
```

表と見くらべてください。Aは65番になっていますね。このアスキーコードの0～31番までは、コントロールコードと言って皆さんが良く使うキャリッヂリターン（13番）や、ベル（7番）など、目に見えない機能が入っているのです。

A$="YURI"の全文字のコードを出したいときは、FOR～NEXTでやればいいワ

じゃあ，もうちょっと手のこんだプログラムを書いてみましょう。先生手伝ってくださいね。

```
10 a$="YURI"
20 for i=1 to len(a$)
30 print right$(a$,i),asc(right$(a$,i))
40 next i
```

実は，これはほとんど先生が書いてくれたんです。自分でプログラムを考えるっていうのはなかなか大変。もっと勉強しなさいですって！　すみません先生。

A$="YURI"でASC(A$)をFOR～NEXTで書けば，I→RI→URI→YURIとなります

さて，実行させたらどうなるかしら。

ちょっと説明しましょう。行番号20では１から始まって A$ の文字数までくり返します。ここでは４ですね。

30番で，RIGHT$（A$, I）と ASC（RIGHT$（A$, I））を表示します。RIGHT$（A$, I）は前にやりましたよね。ストリング変数A$の文字列の右端からＩ個分取れという命令でしたね。ここでＩは，20番により，１から４まで変化することになっています。次に得られたストリングを ASC のカッコ内に入れ，アスキーコードを与えます。

40番は FOR～NEXT ループの NEXT 文です。Ｉが LEN（A$）より大きくなると次の行番号へ移行します。

CHR$(X)はASC関数とは逆の関数で、ASCIIコード番号からキャラクタを選び出すの

さて，アスキーコードを勉強したんですから，もう2つ3つ似たような命令を憶えちゃいましょう。(☞51)

CHR＄（X）

この関数は ASC 関数とは逆に X に入る ASCII コード番号から数字，アルファベット，カナなどのキャラクタを選び出すものだそうです。まあやってみればわかるんじゃないかしら。

32〜255 までを画面に表示してくれます。1〜31 まではコントロールコードといって特殊なコードが入っているため画面には現れて来ないのだそうですよ。

CHR$(X)でA,B,Cを書きましょう!CHR$(X)に数値を代入、たし算で処理できるの

例えば，

```
print chr$(256)
Illegal function call
Ok
■
```

255 を越えるとこんな表示になります。

51

ASC(X)の逆の命令に CHR$(X) という命令があります。たとえば、先程のＡＳＣ("A")は65番でしたね。これを元にもどすと、

```
PRINT CHR$(65)[CR]
A
Ok
▒
```

このように、今度は直接コードから文字を出すことを可能にした命令なのです。

音を出してみましょう。

```
PRINT CHR$(7)[CR]

Ok
▒
```

いかがですか。ピッ！と音が出たでしょう。

```
print "ABC"
ABC
Ok
print chr$(65)+chr$(66)+chr$(67)
ABC
Ok
■
```

上のように，ＡＢＣと書くのに CHR$ を使うと，下のような形になります。こんなコードを用いるのも，いかなる形からでも出力できるということを示すためです。

```
print asc("A")
 65
Ok
print chr$(65)
A
Ok
■
```

ASCやCHR$をわざわざ使うのは、コントロールコードなんかも処理し易いから

このように ASC と CHR$ はまったく逆の命令なんですね。というより裏表の関係と言った方がわかりやすいかしら。

なんで，わざわざこういったコードがつくられているんでしょうか。かえってめんどうな感じがしません？　文字には目に見える文字と見えない文字があるんだそうです。主にこのコードを用いるのは，見えない文字の方。31番までをよく使うんだそうです。昔ならった媒介変数のことを思い出してみてください。例えば，画面を消しなさいとか，そういったコントロールコードの方が使われるわけです。でも文字，数字も一応すべて書き換えられるようにはなっているんですね。

TAB関数を使うと、文の頭の出だしを順々にずらしていくこともできるの

次は文字をあやつる TAB 関数です。

指定された文字を頭から指定されたスペースをとって書き出す命令です。(☞52)

```
print tab(10);"Yuri"
          Yuri
Ok
print tab(5);"Yuri"
     Yuri
Ok
█
```

10個分

5個分

```
10 for i=1 to 10
20 print tab(i);"Yuri"
30 next i
run
 Yuri
  Yuri
   Yuri
    Yuri
     Yuri
      Yuri
       Yuri
        Yuri
         Yuri
          Yuri
Ok
```

52

TAB(X)　タブ関数

現在のカーソル行の一番左からX個のスペースを終了位置まであける命令です。

TAB(X)

終了位置

この終了位置は0から255個までの指定が出来ます。タイプライターのTABと同様に使用することが可能です。

```
PRINT "HUDSON";TAB(1
0);"SOFT"
HUDSON    SOFT
Ok
▒
```

TAB関数では、ずらしていった頭の文字を順次戻していくことも可能なの

このようにスペースが1～10まで10個の YURI を出力してくれました。

このプログラムに少し手を加えてみましょう。

```
10 for i=1 to 5
20 print tab(i);"Yuri"
30 next i
40 for i=5 to 1 step-1
50 print tab(i);"Yuri"
60 next i
```

40番の STEP の命令覚えてますか？　今度はマイナスがついてますねえ。たぶん数字が1つずつ減っていくんでしょうねえ。

さあ実行しますよ。

```
run
 Yuri
  Yuri
   Yuri
    Yuri
     Yuri
     Yuri
    Yuri
   Yuri
  Yuri
 Yuri
Ok
■
```

ずれていく頭の文字を5字ずつにしても、TAB関数はちゃんとそのとおりに…

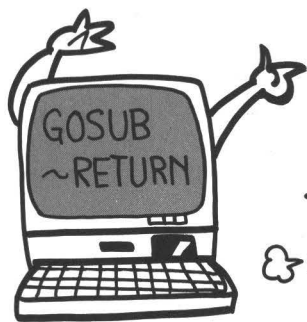
今度は1から35までの TAB を1つずつ増やし，次に1つずつ35から減らして行きます。こんな事も可能です。

```
list
10 FOR i=1 TO 35 STEP 5
20 PRINT TAB(i);"Yuri"
30 NEXT i
40 FOR i=35 TO 1 STEP-5
50 PRINT TAB(i);"Yuri"
60 NEXT i
Ok
■
```

```
run
 Yuri
      Yuri
           Yuri
                Yuri
                     Yuri
                          Yuri
                               Yuri
                                   Yuri
                              Yuri
                         Yuri
                    Yuri
               Yuri
          Yuri
     Yuri
Ok
■
```

5文字

このように STEP を5にすれば5個分のスペースをとって次へ移ってくれます。

こんな風に TAB は書き出す位置を指定し，きれいに見やすしてくれるんですね。

GOSUB文はくり返し部分をプログラムの外に出して、サブルーチン処理にすると

GOSUB 文と RETURN 文の登場です。これは FOR～NEXT のようにワンセットに考えて，サブルーチン処理します。プログラムの中で，同じようなものが何回もでてくる事がしばしばありますが，このような時，くり返す必要のあるものをプログラムの流れの外に出して，繰り返し利用すればとても便利！

この時プログラムの外に出したものをサブルーチン，本すじの流れをメインルーチンと呼ぶのだそうです。(☞53)

このプログラムは，とても単純なものですが，とにかく流れを説明しましょう。

```
list
10 INPUT "ｱﾅﾀﾉ ﾖｿｳ ﾊ";b
20 GOSUB 100
30 IF a=b THEN PRINT "ｱﾀﾘ !" ELSE PRINT
"ﾊｽﾞﾚ.."
40 GOTO 10
100 a=INT(RND(1)*6)+1
110 RETURN
Ok
■
```

まず10番を実行し20番の GOSUB で100番に飛びます。次に110番を実行し，RETURN で GOSUB の次の行，つまり30番へ戻り30，40を実行し，40番でまた10番へ戻る事になりますね。

53
GOSUB RETURN

GOSUBはサブルーチンを呼び出す命令です。このGOSUB文は、GOSUBで呼び出された行からRETURNが書かれている所までを実行します。

RETURNで帰ってくると次の命令を実行します。マルチステートメントの時はGOSUBの次の命令、GOSUBでその行が終っている時は、次の行へいきます。

サブルーチンの中から他のサブルーチンを何回実行してもかまいません。ひとつのRETURN文を複数のGOSUB文で共用してもかまいませんが、サブルーチンの中からGOTOで戻ることはできません。

摂氏・華氏の温度変換プログラムです。GOSUB～RETURNのワンセットでサブルーチン処理

(☞54)

```
list
10 REM *** ｵﾝﾄﾞ C/F ﾍﾝｶﾝ ﾒｲﾝﾙｰﾁﾝ ***
20 CLS:INPUT "ｾｯｼ=c ｶｼ=f (c or f)";a$
30 IF a$="c"THEN 80
40 INPUT "ｶｼﾅﾝﾄﾞ";f
50 GOSUB 1000
60 INPUT "ｵﾜﾘﾏｽｶ (y or n)";b$
70 IF b$="n"THEN 40 ELSE 10
80 INPUT "ｾｯｼﾅﾝﾄﾞ";c
90 GOSUB 2000
100 INPUT "ｵﾜﾘﾏｽｶ (y or n)";c$
110 IF c$="n"THEN 80 ELSE 10
1000 REM ** F ｶﾗ C ﾍ ﾍﾝｶﾝ ｻﾌﾞﾙｰﾁﾝ **
1010 c=(f-32)*(5/9)
1020 PRINT "ｶｼ";f;"ﾄﾞ=ｾｯｼ";c;"ﾄﾞ"
1030 RETURN
2000 REM ** C ｶﾗ F ﾍ ﾍﾝｶﾝ ｻﾌﾞﾙｰﾁﾝ **
2010 f=(9/5)*c+32
2020 PRINT "ｾｯｼ";c;"ﾄﾞ=ｶｼ";f;"ﾄﾞ"
2030 RETURN
Ok
■
```

これは注釈の意味で見出しみたいなものです。コンピュータはREMの入っている行番号をとばし、その次の番号から実行します

これは，セッシ（摂氏）カシ（華氏）の変換プログラムです。このように必要な所を呼び出してくり返すのがサブルーチンの役目。(☞55)

1010
2010 ）変換式

```
ｾｯｼ=c ｶｼ=f (c or f)? c
ｾｯｼﾅﾝﾄﾞ? 25
ｾｯｼ 25 ﾄﾞ=ｶｼ 77 ﾄﾞ
ｵﾜﾘﾏｽｶ (y or n)? n
ｾｯｼﾅﾝﾄﾞ? 0
ｾｯｼ 0 ﾄﾞ=ｶｼ 32 ﾄﾞ
ｵﾜﾘﾏｽｶ (y or n)? n
ｾｯｼﾅﾝﾄﾞ? 100
ｾｯｼ 100 ﾄﾞ=ｶｼ 212 ﾄﾞ
ｵﾜﾘﾏｽｶ (y or n)? ■
```

```
ｾｯｼ=c ｶｼ=f (c or f)? f
ｶｼﾅﾝﾄﾞ? 100
ｶｼ 100 ﾄﾞ=ｾｯｼ 37.777778 ﾄﾞ
ｵﾜﾘﾏｽｶ (y or n)? n
ｶｼﾅﾝﾄﾞ? 0
ｶｼ 0 ﾄﾞ=ｾｯｼ 17.777778 ﾄﾞ
ｵﾜﾘﾏｽｶ (y or n)? ■
```

このように，さっとセッシとカシの変換をやってくれちゃうわけです。

54

REM

これはプログラムの中に注釈を入れるために使用される命令なのです。"ここからは何のルーチン"、"これからはデータです" など、プログラムを作る上でも、チェックする上でも、とても便利な命令です。

REM　’

このREMは、’で置き換えることもできます。

プログラムの実行時には無視されますが、GOTO文やGOSUB文で分岐させることは可能です。

REMの後に：で区切って他の文を続けることはできません。

55

温度計の目盛は世界共通ではありません。日本を含めたアジアの国々ではCセッシ、アメリカ・ヨーロッパではFカシが一般的に利用されております。このセッシとカシの関係は単純な計算では変換ができません。そこで、FからCへの変換式

C=(F-32)*(5/9)

CからFへの変換式

F=(9/5)*C+32

こんな式で展開されるのです。さてこの式を利用してみましょう。

GOSUB～RETURNはつまりこの間でサブルーチンを自由に繰り返されるしるしなの

```
list
10 REM ** ｹｲｻﾝ ﾌﾟﾛｸﾞﾗﾑ **
20 CLS:PRINT "ﾀｼｻﾞﾝ (A+B)=1"
30 PRINT "ﾋｷｻﾞﾝ (A-B)=2"
40 PRINT "ｶｹｻﾞﾝ (A*B)=3"
50 PRINT "ﾜﾘｻﾞﾝ (A/B)=4"
60 INPUT "ﾄﾞﾉｹｲｻﾝ ";k
70 INPUT "A=";a
80 INPUT "B=";b
90 ON k GOSUB 1000,2000,3000,4000
100 PRINT "ｺﾀｴ=";x
110 INPUT "ﾓｳｲﾁﾄﾞｹｲｻﾝ ｼﾏｽｶ (y or n)";a$
120 IF a$="n"THEN END ELSE GOTO 10
1000 REM ** ﾀｼｻﾞﾝ ｻﾌﾞﾙｰﾁﾝ **
1010 x=a+b
1020 RETURN
2000 REM ** ﾋｷｻﾞﾝ ｻﾌﾞﾙｰﾁﾝ **
2010 x=a-b
2020 RETURN
3000 REM ** ｶｹｻﾞﾝ ｻﾌﾞﾙｰﾁﾝ **
3010 x=a*b
3020 RETURN
4000 REM ** ﾜﾘｻﾞﾝ ｻﾌﾞﾙｰﾁﾝ **
4010 x=a/b
4020 RETURN
Ok
■
```

これを RUN させると,

```
ﾀｼｻﾞﾝ (A+B)=1
ﾋｷｻﾞﾝ (A-B)=2
ｶｹｻﾞﾝ (A*B)=3
ﾜﾘｻﾞﾝ (A/B)=4
ﾄﾞﾉｹｲｻﾝ ? 2
A=? 1000
B=? 10
ｺﾀｴ= 990
ﾓｳｲﾁﾄﾞｹｲｻﾝ ｼﾏｽｶ (y or n)? ■
```

さあ GOSUB～RETURN 理解してもらえたかナ？ いかがですか？ メインルーチンの中に GOSUB 文が現われると，その文のあとにくる行番号のサブルーチンに飛び込み，また飛び込んだサブルーチンに RETURN 文が現われると，メインルーチンの，サブルーチンに飛んだときの次の行番号に復帰します。(☞56)

FOR～NEXT との使い方の上での大きな差はループがクロスしてもかまわないし，どこからでも飛び込めるという事。これでプログラムをくり返すのも何の苦労もなくなりましたヨ。

56
もう一つの応用として、四則演算のプログラムを作ってみましょう。これは4つのサブルーチンを用意しました。
このプログラムは、特にサブルーチンにする必要はないのですが、勉強のためにわざと複雑にしたのですよ。

LOCATEって、解析幾何の(X,Y)のような座標位置を指定するための命令なの

```
10 cls
20 locate 10,5
30 print time$
40 goto 20
run
```

5

10 02:23:01

LOCATE という命令は画面上で出力しはじめる位置を指定するものです。LOCATE 10, 5の最初の10はx軸，次の5はy軸の値なんですって。だんだん細かい命令になってきましたけど大丈夫でしょ？この時計は正確に時刻を表示してくれますけど，これを止めるのは

SHIFT + BREAK key のみ (☞57,58)

```
Break in 30
Ok.
■
```

数値をストリングに変換する関数STR$(X)を使うと、数値の合成も簡単ヨ！

STR$ (X) は，数値をストリング（文字列）に変換する関数です。何のためにこんな事する必要があるんだろうと疑問に思うでしょ？ (☞59)

STR$ (X) を使う事によって数値の合成が簡単にできるようになるんですって。そう言われてもあまりピンとこないなあ。

「先生！ 何かプログラムを作ってください」

```
10 print str$(123)
run
 123
Ok
■
```

数値 123

ストリングス123

57
LOCATEは、画面の位置を指定する命令です。
40文字モードでは、
LOCATE(0,0)
から
LOCATE(39,24)
80文字モードでは、
LOCATE(0,0)
から
LOCATE(79,24)
の範囲で使用してください。

58
プログラムを止める時、LISTの途中で止める時には、
SHIFT と BREAK
のキーを同時に押します。
また、
CTRL と C でも同じですよ。(コントロールC)
このコントロールキー CTRL は、いろいろな機能が多く隠されております。
P 252 のコントロールコード表を参考にしてください。

最初の123とSTR$で実行された123は別の性質、実行されて出てきたのは文字だから…

これもプログラムなんだそうですよ。10番の123と実行された123は，まったく別の性質だという事がわかってもらえるかしら？

```
10  a=10
20  b=10
30  print  str$(a*b)
run
 100
Ok
■
```

AとBをかけて、その数をSTR$関数によってストリングとして扱ったもの

ですから，実行されて出て来た100は数値ではなくて，文字なんですよ。

```
10  a=10
20  b=6
30  h$=str$(a*b)
40  print  h$
run
 60
Ok
■
```

A×BをSTR$によってストリングとし、ストリング変数に代入して表示

```
10  print  str$(20)
20  print  str$(0)
30  print  str$(1e+10)
run
 20
 0
 1E+10
Ok
■
```

あれ？　このプログラムの30番は何でしょう？

STR＄は理解したけれど，１ＥのＥって何なのかしら。それに実行されて出て来たものは違っているし……先生，わかりません！　先生ったら「やってなかったっけ……」ですって。どうも，これは指数の問題のようなんですが，教えてもらいましょうネ。

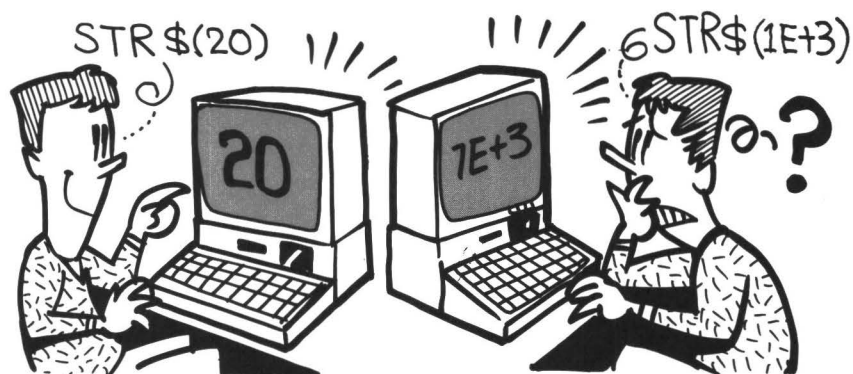
59

STR$は、STRING$とまちがうといけないので、エスティアールダラーと呼びましょう。

STR$(X)

このXは数式、及び数値のみ入れることができます。その数式や数値を、文字列にしなさいという命令なのです。

```
A$=STR$(1000)
Ok
PRINT A$
 1000
Ok
■
```

このように、数字を文字列にしてしまいました。

2E+3は対数と同じようネ! 有効数字が8桁以内だから指数範囲が10^9から10^{38}まで

数値定数としては，符号，数字，小数点からなる10進数と，符号，仮数部，指数部からなる10進数があるんだそうです。

前者の例をあげると +10，−20，0.05，−0.0009 などです。

さっき何だかわからなかったEが登場するのは，後者の方です。

この表示方法は，指数表示といわれ，ある値，たとえば 100 が10の何乗であるかを，一目でわかるように表わしたものなんだそうです。

100 は10^2なので，指数表示を使って表わすと 1E+2 となるんですって。

例　10E−8（0.0000001）
　　仮数部　指数部
　　2E+3（=2000）
　　2.5E+0（= 2.5）
　　2.5E+1（=　25）

対数と同じようだから、宇宙の大きさを表わす天文学的な数も簡単に表示できるの

つまり指数部とは xE のあとについている数値でプラスならば0がふえるわけだし，マイナスなら小数点以下どんどん小さくなるわけです。

対数と同じ考え方よね。対数なんて大嫌いだけど，少しは助けになってくれる事もあるのね。ここで気をつける事は，表現できる数値データに限りがあるという事。有効数字は8けた以内，指数範囲が10^9から10^{38}までですって。いいかしら？

それからもう1つ。コンピュータは原則として仮数を ○.xyzE という風に小数点以下1位から始めるんだそうです。だからさっきのプログラムの1E+10を実行した場合，○.1E+11と表示するコンピュータも当然あるわけ。(☞60)

コンピュータには，この方がオーソドックスな形なんだからね。いろいろな値をEを使って指数表示してみましょうか？

ちょっとばかばかしいけれど，新聞紙を半分に切って重ねて，それを半分に切って重ね，という風に40回繰り返して重ねると，その厚さは？ .165E+12mm ですって。こんなのは？ 人間の心臓が70年間に送り出す血液の量 .184E+9 l。

他には，

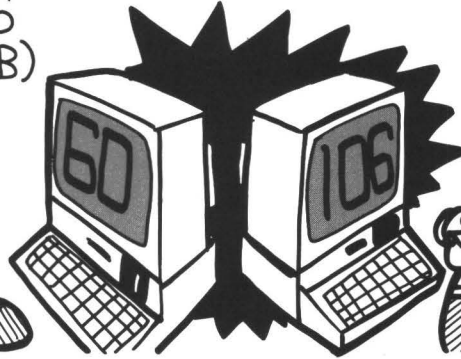
銀河宇宙の直径	.95E+18 km
〃 の総量	.4E+42 kg

などです。

60

Eは Exponential (指数) と言う意味の文字です。

小数点の位置が何個ずれるかということで，3.001E5とは，3.001×10^5で，実は300100のことなのです。

1E2 =100
1E3 =1000
1E4 =10000
1E5 =100000

VAL関数って、STR$(X)と逆にSTRING変数や定数の内容を数値に変えるものなの

ASC(X$), CHR$(X), STR$(X)と出て来たところで **VAL関数**も見ておきましょう。

VAL関数とはSTR$(X)とはまったく逆の関数でストリング変数またはストリング定数の内容を数値にかえるものらしいんです。実際にプログラムを見ていった方が早そうね。

```
10 a$="162"
20 a=val(a$)
30 print a$,a
run
 162           162
Ok
█
```

出力結果を見ると差がないように見えますが，左はストリング，右は数値としてとり扱われています。わかるかなあ？ ではもう1つプログラムを書きましょう。

```
10 a$="12"
20 b$="34"
30 print val(a$+b$)
run
 1234
Ok
█
```

ここで出力された1234は数値であって，けっしてストリングスではありません。

STR$(X)の変数どうしをたして、VALで数値に変換してやれば、数字の合成ができるの

では今度は，数値どうしを合成して新しい数値を作ってみましょう。そのために必要な操作はというと……例として12と34を使って 1234 という新しい数値を作りましょう。(☞61)

```
10 a=12
20 b=34
30 a$=str$(a)
40 b$=str$(b)
50 c$=a$+b$
60 c=val(c$)
70 print c
run
 1234
Ok
█
```

行番号 10,20　数値をそれぞれの数値変数に代入

〃　30,40　数値をストリングに変換して，それぞれのストリング変数に代入

〃　50　ストリング変数のたし算

〃　60　ストリング変数を数値に変換して数値変数に代入

〃　70　数値変数Cの内容を出力

どうですか，このプログラムには，STR$(X)とVAL(X$)の両方が使われています。理解してさえいれば，難しくないプログラムのはずなんだけど，どうかナ？

61

VAL(X$)

バリューはSTR$の逆の関数です。

```
PRINT STR$(10)
 10
Ok
PRINT VAL("10")
 10
Ok
▒
```

文字列を数字に置き換えるのです。

```
A$="123"
Ok
A=VAL(A$)
Ok
PRINT A
 123
Ok
▒
```

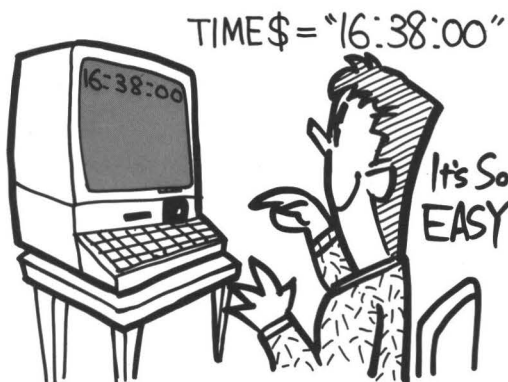
ほら、この様に数字変数に取り込まれたでしょう。それではいろいろ試してみてくださいね。

STR$(X)のたし算はならべて書くという意味だから、かけ算に変えてもだめなのです

ここで，50番の＋を*にしてみたらどうかしらなんて考えていたんだけど，どう思う。

```
10 a=12
20 b=34
30 a$=str$(a)
40 b$=str$(b)
50 c$=a$*b$
60 c=val(c$)
70 print c
run
Type mismatch in 50
Ok
■
```

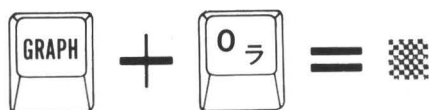
やっぱり*じゃだめなんですね。考えてみれば当然の結果かな……。だって，50番の段階ではストリング変数だから，たし算だけ，というよりはならべて書くだけしかできないわけよね。

＋の記号は使っていても，文字と文字をたすことなんてできるはずないから，この場合は，ならべて書きなさいという記号とでも理解した方がいいでしょう。かけ算しなさいなんて，無理な注文というもの。60番の VAL ではじめて数値となるんですもの。ですから VAL (X$) で X$ の内容は，ストリング定数，ストリング変数及びストリングのたし算のどれかになるわけです。

画面上に1秒ごとに数字の変わるデジタル時計を出しましょう！現在時刻の指定も必要

基本的命令は，もうかなり出てきましたネ。皆さんも，そろそろ，大きなプログラムの1つでもと思ってるんじゃないかしら。

そこで，先生と一緒に，時計でも作ってみようという事になりました。画面上に1秒ごとに数字のかわるデジタル時計が出てくるんですよ，これは楽しみ。では忍耐を重ね，とにかく，あきらめずに作りあげたプログラムを見てください。あっ！その前に，必ず時間を指定して，コンピューターのタイマーと時刻を合わせておいてくださいね。ただ TIME$="16:38:00"という風に入力するだけ。必ず6けたの数字でね。（☞62）

入力した時は4時38分0秒だったんですね。もしこれを忘れると，コンピュータの電源を入れてからの時間が表示されちゃうんですよ。どうぞご注意！

画面いっぱいに出る時間表示の数字はGRAPHキーを使って打ち込みましょう。

62
タイマーの設定

X1には年月日、時分秒を記憶表示することができる機能があります。一度設定することにより、内蔵充電式電池があるかぎり動作しておりますよ。

DATE$="82/11/15"

日付の設定
かならず／で区切ります。

TIME$="16:38:40"

時間の設定
かならず：で区切ります。

時計表示は配列変数S$が或る範囲で変化するという考え方が基本になっているの

```
time$="16:38:00"
Ok
■
```

配列変数で変数の範囲を指定します。この場合は2次元ストリング配列です。

```
list
10 DIM s$(10,7)
20 CLS
30 FOR j=0 TO 9:FOR i=1 TO 7:READ s$(j,i
):NEXT i,j
40 PRINT:PRINT"┌──────────────────────
───────────┐"
50 FOR i=2 TO 19:LOCATE 0,i:PRINT "|":NE
XT i
60 FOR i=2 TO 19:LOCATE 38,i:PRINT "|":N
EXT i
70 LOCATE 0,20:PRINT "└──────────────────
───────────┘"
80 LOCATE 19,4:PRINT "◆":LOCATE 19,8:PRI
NT "◆"
90 LOCATE 3,15:PRINT "DIGITAL CLOCK"
100 LOCATE 3,17:PRINT "Yuri Shinagawa"
110 h1$=LEFT$(TIME$,1):h1=VAL(h1$)
120 h2$=MID$(TIME$,2,1):h2=VAL(h2$)
130 m1$=MID$(TIME$,4,1):m1=VAL(m1$)
140 m2$=MID$(TIME$,5,1):m2=VAL(m2$)
150 s1$=MID$(TIME$,7,1):s1=VAL(s1$)
160 s2$=RIGHT$(TIME$,1):s2=VAL(s2$)
170 FOR i=1 TO 7
180 LOCATE 5,2+i:PRINT s$(h1,i):LOCATE 1
2,2+i:PRINT s$(h2,i)
190 LOCATE 23,2+i:PRINT s$(m1,i):LOCATE
30,2+i:PRINT s$(m2,i)
200 LOCATE 23,11+i:PRINT s$(s1,i):LOCATE
 30,11+i:PRINT s$(s2,i)
210 NEXT i
220 GOTO 110
1000 DATA"       "
1001 DATA"  ▒▒▒  "
1002 DATA" ▒   ▒ "
1003 DATA" ▒   ▒ "
1004 DATA" ▒   ▒ "
1005 DATA" ▒   ▒ "
1006 DATA" ▒   ▒ "
1007 DATA"  ▒▒▒  "
2001 DATA"  ▒▒   "
2002 DATA"   ▒   "
2003 DATA"   ▒   "
2004 DATA"   ▒   "
2005 DATA"   ▒   "
2006 DATA"   ▒   "
2007 DATA"  ▒▒▒  "
3001 DATA"  ▒▒▒  "
3002 DATA" ▒   ▒ "
3003 DATA"     ▒ "
3004 DATA"  ▒▒▒  "
3005 DATA" ▒     "
3006 DATA" ▒     "
```

と，こうなっちゃいます。

このプログラムを行うごとに説明すると，

10…配列変数で変数の範囲を指定します。この場合は２次元ストリング配列です。

20…一度画面を消しましょうという命令でしたね。

30…Ｊが０から９まで，Ｉが１から７までの間で次々に作られる。（Ｊ，Ｉ）を読み取るのです。

50…Ｉが２から19までは（０，Ｉ）の座標上に｜を書くことをくり返すワケです。

70…（０，20）の場所から└─── を書き始めます。

80…（19，4）に◆，（19，8）に◆をそれぞれ書きます。

90…（3，15）から DIGITAL CLOCK を書き出します。

100…やはり（3，17）の点から Yuri Shinagawa を横に表示していってくれます。

110…これは何かというと，○○：○○：○○という表示。つまり TIME$ の1番左はしの1文字をとり，それを数値として表示せよという意味です。

120…110番と同様に今度は MID を使って，TIME$ の途中，つまり左から2文字目から1文字分を数値に変えて表示しています。

160…今度は右はしから1文字分ですね。

170…Iの範囲を指定します。

180…左から2文字目まで，つまり○○時の部分の場所を指定して表示させます。

190…同様に○○分の部分を定めます。

200…同様に○○秒の部分です。

210…170行の FOR を受けます。

220…時計ですから数値はドンドン変わりますね。110行にもどり，何回も同じことをくり返すわけです。

1000…1000番以下は表示される数字の実際の形のデータです。ここから始めなければならないのが，めんどうといえばめんどうなところ。

最後にプログラミングをスッキリと整えて、My Clockの完成ですよ!

さて，こんなかわいい時計ができちゃいました。うれしいなあ! でも喜んでばかりはいられないのです。このプログラム，もっと省略できる部分はないかしら？ どうでしょうか先生？ えっ？そうですか。どうやら110番〜160番はもう少しきれいな形になるみたいですよ。

```
110 h1$=left$(time$,1):h1=val(h1$)
```

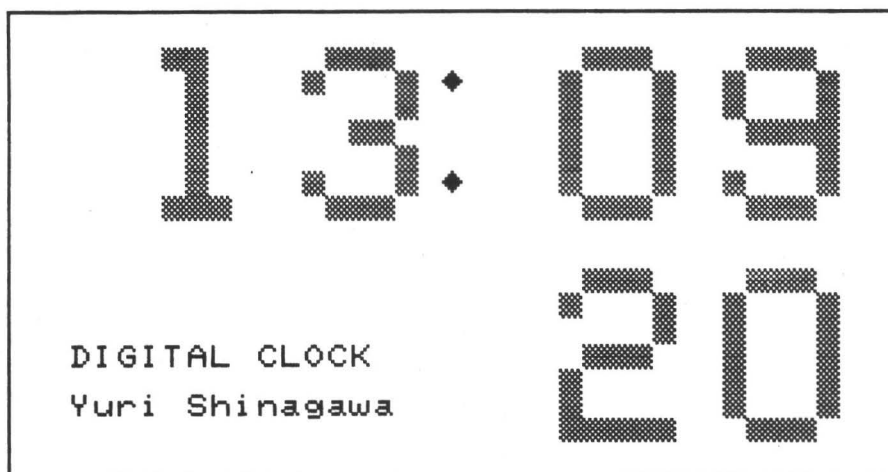
そうねえ。よく見ると＝が２つも使ってあって，ちょっとまわりくどい書き方ですねえ。

そこで，

```
110 h1=val(LEFT$(TIME$,1))
120 h2=val(MID$(TIME$,2,1))
130 m1=val(MID$(TIME$,4,1))
140 m2=val(MID$(TIME$,5,1))
150 s1=val(MID$(TIME$,7,1))
160 s2=val(RIGHT$(TIME$,1))
```

と書き直してみました。これでいいわけですね。

やっとできた私の傑作!! 絶対テープにとっておかなくちゃね!

```
save"Digital clock"
Writing"Digital clock.    "
Ok
▒
```

コンピュータの話とはまったく関係ないんですけど，ちょっと目についた記事があるんですよね。

「男性と女性では脳の使い方が違う」こんなタイトルがついてたんだけど，本当なのかしら？　どう思います？

男性と女性がいろいろな面で違うというのは，当然の事ですけど，科学者の中には体の大きさとか，体力とかいう生理的な面以前に，もっと基本的な相違があると考える人もいるそうです。

どうやら，男と女では，外界の事象を経験する仕方にしても違うらしいんです。男女はそれぞれ触覚も聴覚も異なり，何か問題を解くときでも，異なった脳細胞を働かせているようです。

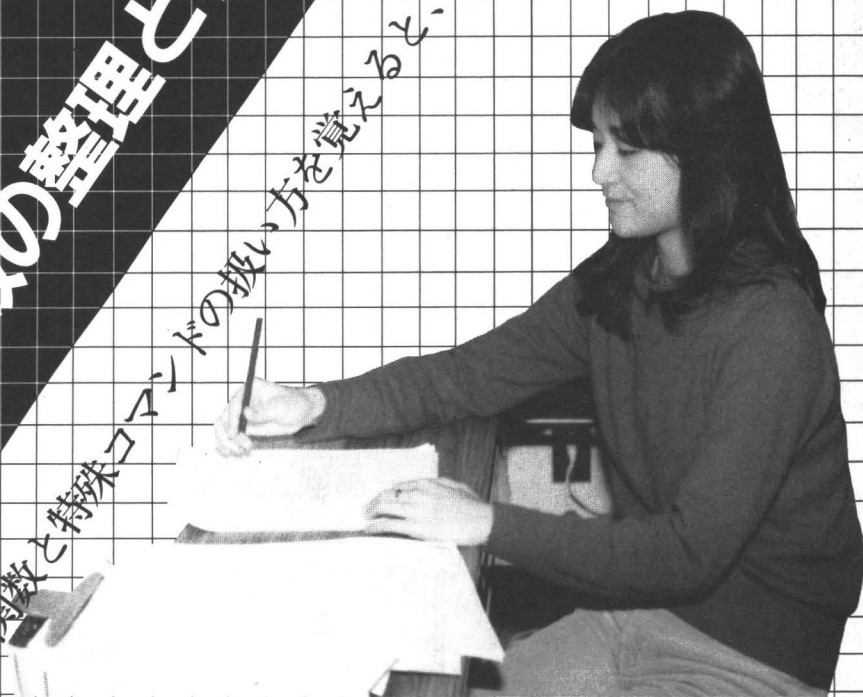
でも，そう言われてみれば，女の子って一般的に数学とか弱いし，そのかわり語学はけっこう強かったりするでしょ？

男性の脳の組織的特徴は，視覚による空間認知能力が優れている事なんですって。数学に強いのもうなづけますねえ。

それに，女性の脳の方が言葉を扱うのに適している事も，ある程度明らかにされているようです。

本当に男の人と女の人とでは，脳の使い方が違うのかしら……？　長い間の慣習によりそうなったんじゃないのかしら？後天的なものだと考えてる学者もずいぶんいるようだし……。

先天的なものだなんていわれたら，どうしようもないものねえ。せっかくコンピュータをやってみようかなぁーなんて考える女の子が増えてきているのに，生まれつき男の子よりも数字に弱いんだ，なんて言われたらショックよねぇ。(P236に続く)

NOTE6
関数の整理と特殊コマンドの勉強
関数と特殊コマンドの扱い方を覚えると、プログラミングの世界が大きく拡がる

もうすでに INT（インテジャー）とか，RND（ランダム）などの数値関数は出てきていますが，ここでまとめてとりあげておきましょう。関数というと，どうしても三角関数のイメージがあるんだけど，さて，どんな関数が出てくるでしょうか？

ABS(X)は絶対値の計算関数、( )内には数値変数、数値定数、計算式が入ります

ABS（X）（アブソリュート）は絶対値の計算をしてくれる関数です。ABS というのは absolute value の略だそうで……，

（☞63）

```
PRINT ABS(-6)
 6
Ok
▒
```

```
PRINT ABS(6)
 6
Ok
▒
```

```
PRINT ABS(-.5)
 .5
Ok
▒
```

（ ）の中には数値変数，数値定数，計算式が入ります。ストリングはダメですから気をつけて！

63
関数　a function

関数とは何でしょうか。そうですね、一般的に使われているのが三角関数という言葉ですが、それだけではありません。数字の ある 数が ほか の数の変化にしたがって変化するとき、ほか の数に対する ある 数の呼び名を関数と言うのです。

ですから、色々な数の変化、文字の変化を行うことができる命令を、ここでは関数と言っております。関数の数々を、ゆりちゃんのノートで覚えてください。

三角関数SIN(X), COS(X), TAN(X)は(　)内の度数をラジアン単位に変換して使います

さて，次は**三角関数**です。**SIN**（サイン），**COS**（コサイン），**TAN**（タンジェント）は，皆さん知ってますよね。SIN（X），COS（X），TAN（X）のXには，定数や計算式が入りますけれど，気をつけなくてはならない事があるんです。

数Ⅰの教科書にも書いてあったけど，度数表示の問題です。Xは常にラジアンで表さなければならないんです！

度をラジアンにかえる変換式を書いておきましょう。

X(ラジアン)＝A(度)＊pai(1)/180

pai(1)は3.14……のこと。pai(2)は6.28……，pai(3)は9.42……となります。おわかりカナ……？

さて，45度のSINを求めてみますよ。まず45度をラジアンに変換します。

```
X=45*PAI(1)/180
Ok
PRINT X
 .78539816
Ok
▒
```

ラジアンが求まりました。さあSINを出しますよ。

```
PRINT SIN(.78539816)
 .70710678
Ok
▒
```

せっかくラジアンを求めてあるのですから，COS，TANも求めちゃいましょうね。

```
PRINT COS(.78539816)
 .70710678
Ok
▒
```

え〜と。sin 45° は$\frac{1}{\sqrt{2}}$で，cos 45° も$\frac{1}{\sqrt{2}}$だから，同じ結果でいいのよね。

```
PRINT TAN(.78539816)
 .99999999
Ok
▒
```

フーン！　なるほど。TAN 45°は1と考えますけど，本当は限りなく1に近づくのだから，これが正しいんでしょうね。

ATN(X)はTAN(X)を逆にしたものだから、ATN(X)＝TAN^{-1}(X)になるの

もうひとつ，**ATN**（X）（アークタンジェント）も勉強しましょう。

ATN(X)＝TAN^{-1} (X)

TAN 45度は1ですけど，TAN^{-1} を使えば1から45度が算出できるんですよ。

ATN の結果は，ラジアンで出てきますから，今度はラジアンから度に変換する事が必要となってきます。

```
PRINT ATN(1)
 .78539816
Ok
▒
```

A(度)＝X(ラジアン)＊180/pai(1)

さて，度に直すのには……。

```
PRINT .78539816*180/PAI(1)
 45
Ok
▒
```

これでＯＫ！

次はXの値を倍精度に変換する **CDBL**（X）（コンバートダブル）命令です。

```
PRINT CDBL(1.15)
 1.1499999999906868
Ok
▒
```

なる程ね。倍精度ですから16けたまで表示してくれるのネ。

さらに **CSNG**（X）（コンバートシングル）は，CDBL とは逆にXの値を単精度に変換します。

```
PRINT CSNG(1.15)
 1.15
Ok
▒
```

```
PRINT CSNG(1.156784321695421)
 1.1567843
Ok
▒
```

8けた（単精度）よりも多いけた数ならば，このように，8けたまで表示してくれます。

LOG(X)はXの自然対数を与えるコマンド、逆にEXP(X)は自然対数の底とのX乗を関数値に…

LOG（X）は，Xの自然対数を与える命令です。

```
PRINT LOG(2)
 .69314718
Ok
▒
```

```
PRINT LOG(-1)
Illegal function call
Ok
▒
```

あれ？ LOG を使う時に注意する事は，Xの値なんです。常にXはゼロより大きくなければなりません。

EXP（X)はイクスポーネンシャルと言って，log とほぼ逆のようなもの。自然対数の底とのX乗を関数の値として持ちます。

```
PRINT EXP(.69314718)
 2
Ok
▒
```

ほらね！ log（2）と比べてごらんなさい。

SGN(X)はX＞0,X＝0,X＜0でそれぞれ1,0,－1の符号を与えるコマンドなの

SGN（X）は，Xに符号を与える命令です。

X＞0　→1

X＝0　→0

X＜0　→－1

Xは1，0，－1の3つのどれかに必ず分けられてしまいます。

```
PRINT SGN(9)
 1
Ok
PRINT SGN(0)
 0
Ok
PRINT SGN(-4)
-1
Ok
▒
```

わかりますよね。

SQR（X）（スクウェアルート）は，Xの平方根（$\sqrt{X}$）を求める命令です。スクウェアルートという名前からも，どんな命令か推測できますネ！

```
PRINT SQR(4)
 2
Ok
PRINT SQR(2)
 1.4142136
Ok
▒
```

平方根なんて何年ぶりかなあ。文科系だとホントに接する機会がないものだから，すぐ忘れちゃうのよネ。でも忘れても大丈夫コンピュータにおまかせです。

SUM(X)は、1からXまでの和を合計してしまう命令なの、便利ですヨ!

SUM（X）（サム）は，1からXまでの和を，パッと出してくれる便利な命令です。1＋2＋3＋……なんて，もうやらなくていいんですヨ！

```
PRINT SUM(10)
 55
Ok
PRINT SUM(100)
 5050
Ok
PRINT SUM(1000)
 500500
Ok
▒
```

FIX（X）（フィックス）……さて，これはXの整数部のみをとり出し，小数点以下を切り捨てる命令です。四捨五入は行ないませんから注意してネ。

```
PRINT FIX(3.6578)
 3
Ok
▒
```

こんなふうになるんだけれど，この命令とよく似たものを，どこかでやったような気がしません？　ここで是非思い出して欲しいのがFRAC（X），フラクションです。FIX（X）が整数値だけをとり出し，小数点以下を切り捨ててしまうのに対し，FRAC（X）は，小数点以下のみをとり出す命令でしたね。

```
PRINT FRAC(3.6578)
 .6578
Ok
▒
```

今まで出て来た関数の中で，このように裏表のような関係のPairが，けっこうありましたよね。こういうものは，一緒に覚えてしまうと，理解は速いし，また深くもなります。

関数を上手に使って，プログラムを楽しみましょう。

今まで，随所に出て来た IF～THEN 文などは，**分岐文**と呼びます。これがプログラムの中に増えると，すごく難しくなってきちゃいます。

頭の中で流れをしっかり考えて，プログラムを組んだつもりでも，わからなくなるようなことはよくあるんです。

だから，プログラムを作る時は，流れ図を書いてください。流れ図の事を**フローチャート**と言います。

フローチャートで使う記号はIF文、PRINT文、GOSUB文などそれぞれ異るのです

フローチャートを作成するのに使う，フローチャート記号の説明をしておかなくちゃネ！(図１)

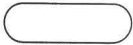
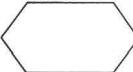
端子

プログラムの始まり，終わりの記号

準備

定義文など，変数値の設定もこの記号

処理（代入文）

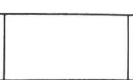
あらゆる種類の処理機能に使えます

手操作入力（INPUT 文，READ 文）

キーボードから入力する時の記号

判断（IF 文）

分岐文の中でどの経路へ進むかを決める記号

書類

図１

端子
プログラムの始まり、終わりの記号

準備
定義文など、変数値の設定もこの記号

処理（代入文）
あらゆる種類の処理機能に使えます

手操作入力
(INPUT文、READ文)
キーボードから入力する時の記号

判断（IF文）
分岐文の中で、どの経路へ進むかを決める記号

書類
データの内容をPRINT OUTする記号

表示（PRINT文）
ディスプレイ上にデータをPRINT OUTする記号

定義済み処理（GOSUB文）
プログラム中に定義してあるサブルーチンに飛ぶことを示します。
㊟サブルーチンとは、プログラムの流れの外に出した副プログラムのようなものでしたね。

データの内容を PRINT OUT する記号

表示（PRINT 文）

ディスプレイ上にデータを PRINT OUT する記号

定義済み処理（GOSUB）

プログラム中に定義してあるサブルーチンに飛ぶことを示します

㊟サブルーチンとはプログラムの流れの外に出した副プログラムのようなものでしたよね

結合子

フローチャートが長くなり結べない時などに使います

ON〜GOTO 文，ON〜GOSUB 文にも使います

こういう図を使うと，プログラムを書くのが楽しくなってきそうですねぇ。

さて，クイズです。左の図の名称を右から選んで結んでみてください。2つ以上間違えたら不合格！　前のページにもどって勉強し直しましょう。(図2)

```
LIST
10 FOR X=1 TO 9
20 FOR Y=1 TO 9
30 PRINT X*Y;
40 NEXT Y
50 PRINT
60 NEXT X
Ok
```

さて，このプログラムをフローチャートにしてみましょう。(図3)のようになりました。ところで，このプログラム何だか憶えてる？　そうです。九九の計算プログラムでしたよネ。

```
RUN
 1  2  3  4  5  6  7  8  9
 2  4  6  8  10  12  14  16  18
 3  6  9  12  15  18  21  24  27
 4  8  12  16  20  24  28  32  36
 5  10  15  20  25  30  35  40  45
 6  12  18  24  30  36  42  48  54
 7  14  21  28  35  42  49  56  63
 8  16  24  32  40  48  56  64  72
 9  18  27  36  45  54  63  72  81
Ok
```

今度は4までの数当てゲームのプログラムに挑戦してみます。

結合子
フローチャートが長くなり結べない時などに使います。ON〜GOTO文、ON〜GOSUB文にも使います。

図2

○手操作入力

○表示

○定義済み処理

○端子

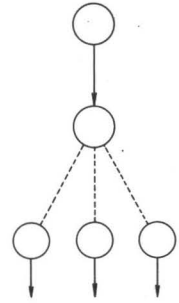
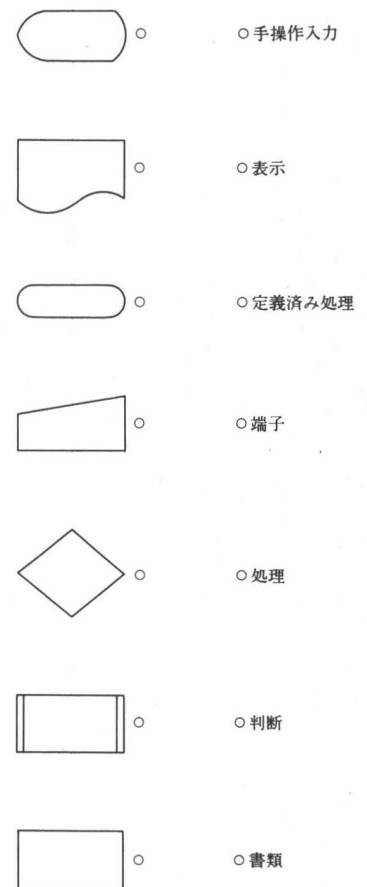
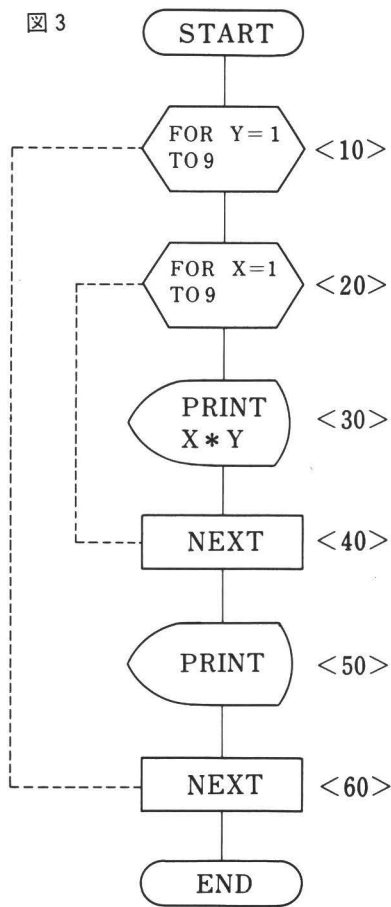
○処理

○判断

○書類

図3

㊟<　>は行番号

図4

```
10 CLS
20 A=INT(4*RND(1))+1
30 IF A<2 THEN PRINT "END":END
40 PRINT "Ok"
50 GOTO 20
```

さてフローチャートにしてみました。(図4)

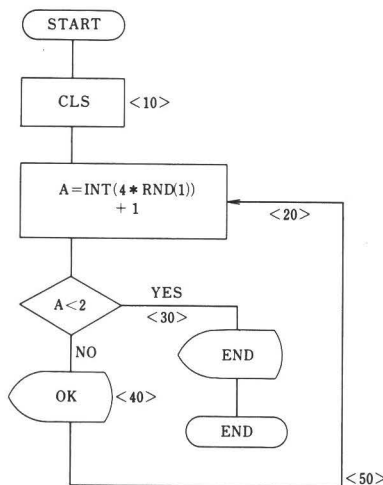
○10番は画面を消す処理です。

○20番は代入文ですね。

○30番は判断文。ここでYES，NOに分岐しています。ここでYESならENDと表示して，プログラムは終了。NOなら40番へ行きます。

○40番で OK と表示。

○50番で20番へ戻ります。

サブルーチンの特徴は煩雑な流れの処理をメインルーチンの流れの外に出せること

```
LIST
10 CLS
20 A=INT(4*RND(1))+1
30 INPUT B
40 IF A=B THEN PRINT"ｱﾀﾘ" ELSE PRINT "ﾊｽ
ﾞﾚ.."
50 GOTO 20
Ok
```

今度は，30番に INPUT 文が入ってますね。えーと……，INPUT 文は手操作入力になるわけかな……。

それでは，フローチャートを見てみましょう。図にしてみると，わかりやすいわネ。では実行させますよ。(図5)

```
? 2
ハズレ・・
? 3
アタリ
? 4
ハズレ・・
? ▒
```

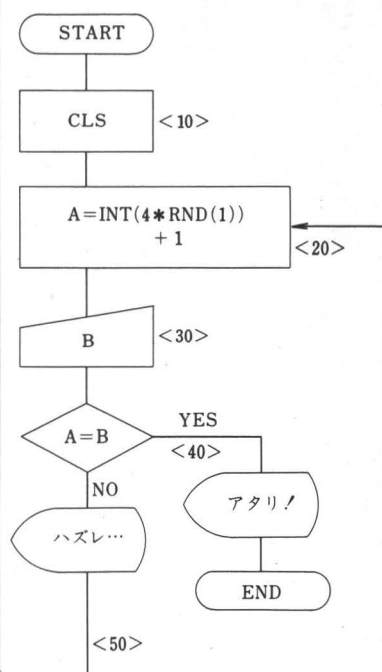
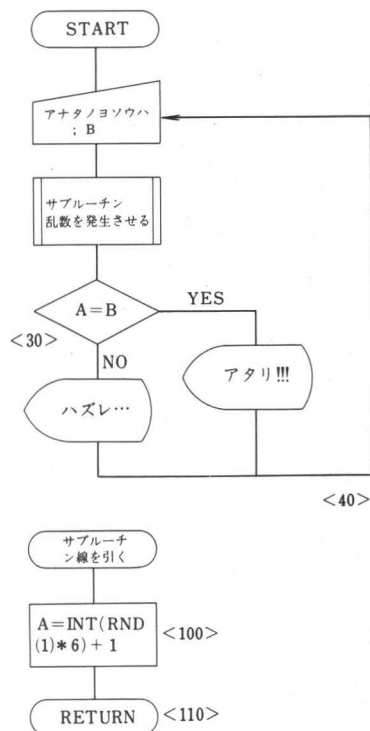
今度は，GOSUB〜RETURN 文の使われたプログラムを，フローチャートで描いてみましょう。(図6)

```
LIST
10 INPUT "アナタノ ヨソウ ハ ";B
20 GOSUB 100
30 IF A=B THEN PRINT "アタリ !" ELSE PRINT
"ハズレ.."
40 GOTO 10
100 A=INT(RND(1)*6)+1
110 RETURN
Ok
▒
```

サブルーチンの処理は，このように流れの外に出して，副プログラムという形で扱うんです。わかりました？

本当は，この程度の優しいプログラムならフローチャートは書かなくてもいいようなものなんですって。難しくってこみ入ったプログラムになればなる程，フローチャートは意味が出てきますよ。

図5

図6

LINEINPUT

```
list
10 PRINT "アナタ ノ ナマエ ハ ";
20 LINEINPUT  a$
30 PRINT a$;"サンテ゛ス。"
Ok

run
アナタ ノ ナマエ ハ シナカ゛ワ
アナタ ノ ナマエ ハ シナカ゛ワサンテ゛ス。
Ok
■

10 INPUT "アナタ ノ ナマエ ハ ";a$
20 PRINT a$;"サンテ゛ス。"
run
アナタ ノ ナマエ ハ ? シナカ゛ワ
シナカ゛ワサンテ゛ス。
Ok
■
```

一行すべてを入力してくれる命令です。2つのプログラムを見比べてみてください。上の方は，RUN すると10番，20番と“サンデス”をつづけて一行で書いてくれます。アナタノナマエハ？の？が出てこないのも INPUT との差！（☞64）

でも，不便な点もあって，必ずストリングでなければ，LINE-INPUT できません。まあ，場合により，うまく使いわけてくださいネ。

LPRINT

```
lprint "HUDSON"
print "HUDSON"
HUDSON
Ok
■
```

> 64
>
> LINEINPUT X$
>
> この命令は、読んで字のごとく、今カーソルのある一行すべての文字を取り込みなさい、という命令です。

直接プリンタに，表示を出しなさいという命令です。ですから，PRINT の内容はディスプレイには現れませんよ。実務用には使い道の広い命令です。

PRINT USING

Print using "###. ##" ; a

こんな形で使います。前もって指定されたaは 12345. 12345 としましょう。(☞65)

```
a=12345.12345
Ok
print using "#####.##";a
12345.12
Ok
print using "######";a
␣12345
Ok
print using "Total=#####.#####";a
Total=12345.12345
Ok
print using "####.####";a
Format over
Ok
▒
```

指定が5桁なのでひとつスペースがあきます。

プログラムを見てもらえば分かるのですが，整数部は今5桁と指定されているのに，# を4桁で出力させようとするとエラーが出てしまいました。この Print Using を使用する時には，あらかじめ桁数が増えることを見越して，余裕を取っておいてください。

このように，ストリングの中に #以外のものを入れれば，それはそのままプリントアウトされます。

```
print using "#####.##";12345.67
12345.67
Ok
print using "Yuri ##.##";12.34
Yuri 12.34
Ok
▒
```

また上のように，ストリングスのあとに直接セミコロンでつなげて定数を指定しておいてもいいわけです。こんな説明しかできないんですけど，あとはプログラムとにらめっこしながら，ゆっくり考えてくださいネ!

65

USING 書式指定

PRINT USING

による書式指定は、数値型と文字型の2種類があります。

①数値型

#　指定した最大の桁数より数値の桁数が小さいときは、右づめで表示されます。

```
PRINT USING "######"
;2345
␣␣2345
Ok
▒
```

.　固定小数点の表示で、小数点の位置をそろえる指定には・を使います。小数点以下の桁指定も行えます。

```
PRINT USING "###.##"
;23.456
␣23.46
Ok
▒
```

この指定では、小数点以下第2位で四捨五入されます。

,　数値を3桁ごとに区切って、そこに、カンマを入れて右づめで表示します。

```
PRINT USING "###,###
,###";23456789
␣23,456,789
Ok
▒
```

+と−　+と−は、数値を表示するときに、+と−の符号をどこに表示するか指定します。

```
PRINT USING "####+",
1234
1234+
Ok
PRINT USING "-####",
1234
-1234
Ok
▒
```

******　桁の指定をして、その範囲内でスペースがある場合、すべて*でうめるという指定をします。

```
PRINT USING "*******
";234
****234
Ok
▒
```

¥¥　数字の前に¥マークをつけてくれます。

```
PRINT USING "¥¥#####
";234
    ¥234
Ok
▒
```

****¥**　**と¥¥を組み合せたものです。数字の前に¥をつけて、その範囲内でスペースがある場合、*でうめます。

```
PRINT USING"**¥####"
;234
***¥234
Ok
▒
```

^^^^　　桁数指定の#の後に置くとエキスポーネンシャルにて表示できるようになります。

```
PRINT USING "##.###^
^^^";234
 2.340E+02
Ok
▒
```

1つのUSING文で複数の変数、実数を記述すると、その変数、実数すべてに対し、USING指定したとみなされます。

```
PRINT USING "####.##
",23.4;102.456
  23.40 102.46
Ok
▒
```

②文字型

!　　　　これを指定すると、文字変数、文字列にある文字の最初の1文字のみ表示します。

```
PRINT USING "!";"ABC
"
A
Ok
A$="XYZ"
Ok
PRINT USING "!";A$
X
Ok
▒
```

&　&　　指定文字列を&と&のスペースの個数+2個分表示します。文字は、左づめで、スペースの個数+2個以下のときは残りの右の部分をスペースでうめます。

```
A$="HUDSON SOFT"
Ok
PRINT USING "&       &
";A$
HUDSON
Ok
PRINT USING "&
       &";A$
HUDSON SOFT
Ok
▒
```

③その他

USINGの書式指定以外の文字がある場合は、それをそのまま表示します。

```
PRINT USING "YURI ##
#",150
YURI 150
Ok
PRINT USING "###.##
sec",123.45
123.45 sec
Ok
▒
```

KEY コマンド

```
KEY 1,"AUTO"+CHR$(13)
KEY 2,"?TIME$"+CHR$(13)
KEY 3,"KEY"
KEY 4,"LIST"+CHR$(26,13)
KEY 5,"RUN  "+CHR$(13)
KEY 6,"LOAD  "+CHR$(13)
KEY 7,"WIDTH "
KEY 8,"CHR$("
KEY 9,"PALET "
KEY10,"CONT"+CHR$(13)
Ok
█
```

ファンクションKEYを使う命令です。ファンクションKEY 5つと SHIFT +ファンクションKEYで，計10個のKEYがあると考えられます。その1つ1つに，よく使うコマンドなどを入れておくことができるんです。いちいちKEYをたたく命令が省けちゃうというわけ。(☞66)

```
key 1,"INPUT"
Ok
KEYLIST

KEY 1,"INPUT"
KEY 2,"?TIME$"+CHR$(13)
KEY 3,"KEY"
KEY 4,"LIST"+CHR$(26,13)
KEY 5,"RUN  "+CHR$(13)
KEY 6,"LOAD  "+CHR$(13)
KEY 7,"WIDTH "
KEY 8,"CHR$("
KEY 9,"PALET "
KEY10,"CONT"+CHR$(13)
Ok
█
```

このプログラムを見てください。このように，10個まで入れておけるでしょう。電話でいえば，短縮ダイヤルというところかしら……？

```
key 1,"YURI"+chr$(13)
Ok
KEYLIST

KEY  1,"YURI"+CHR$(13)
KEY  2,"?TIME$"+CHR$(13)
KEY  3,"KEY"
KEY  4,"LIST"+CHR$(26,13)
KEY  5,"RUN  "+CHR$(13)
KEY  6,"LOAD  "+CHR$(13)
KEY  7,"WIDTH "
KEY  8,"CHR$("
KEY  9,"PALET  "
KEY10,"CONT"+CHR$(13)
Ok
YURI
Syntax error
Ok
■
```

ところで，上のプログラムを見てみましょう。KEY 1 を押したら，Syntax error が出てしまいました。どうしてかなあ…？

結局，KEY 1 の CHR$(13) というのは余分なんですね。ただ YURI と入れてあるだけで PRINT 命令もしてないわけだから，キャリッヂ・リターンを示す CHR$ (13) は不用！

じゃあ，KEY 1 を書き直しましょう。

```
key 1,"print yuri"+chr$(13)
Ok
KEYLIST

KEY  1,"print yuri"+CHR$(13)
KEY  2,"?TIME$"+CHR$(13)
KEY  3,"KEY"
KEY  4,"LIST"+CHR$(26,13)
KEY  5,"RUN  "+CHR$(13)
KEY  6,"LOAD  "+CHR$(13)
KEY  7,"WIDTH "
KEY  8,"CHR$("
KEY  9,"PALET  "
KEY10,"CONT"+CHR$(13)
Ok
print yuri
 0
Ok
■
```

ここでは，CHR$(13) はしっかり意味を持ってきますね。ただまだ KEY 1 には困った点があるんです。KEY 1 を実行した時，YURI は変数扱いになっています。これを文字列で扱わせるには……？

66

電源打入時のファンクションキーの設定

電源打入時は、左上KEYLIST の様に設定されていますが、この内容は任意に変更することができます。

なお、CHR$(13)はキャリッジ・リターン命令で、このKEYだけでCRまで完了してしまうのです。

```
key 1,"print"+chr$(34)+"Yuri"+chr$(13)
Ok
KEYLIST

KEY 1,"print"+CHR$(34)+"Yuri"+CHR$(13)
KEY 2,"?TIME$"+CHR$(13)
KEY 3,"KEY"
KEY 4,"LIST"+CHR$(26,13)
KEY 5,"RUN  "+CHR$(13)
KEY 6,"LOAD "+CHR$(13)
KEY 7,"WIDTH "
KEY 8,"CHR$("
KEY 9,"PALET "
KEY10,"CONT"+CHR$(13)
Ok
print"Yuri
Yuri
Ok
■
```

さて次はこのプログラムですよ。ほらね。こんな風に CHR $ (34) を入れておけば，きちんとストリング扱いをしてくれるんです。CHR $ (34) は，ダブルクォーテーションです。

でも，ここではダブルクォーテーションが，最初だけしかついてないわね。これでも大丈夫なのかしら？

```
print "Hudson
Hudson
Ok.
print "Hudson:a=1
Hudson:a=1
Ok.
print "Hudson":a=1
Hudson
Ok.
■
```

なる程，後に何かつづける時以外は，後の方の ” は省略できますからね。

INKEY $

変数 A$ の中に何か一文字分とり込みなさいという命令です。ですから一文字のみの判断で，プログラムが進んでいきます。

```
list
10 a$=INKEY$
20 PRINT a$;
30 GOTO 10
Ok
■
```

ちょっとこのプログラムを見てください。

```
10 a$=inkey$:if a$="" then 10
20 print a$
30 goto 10
```

RUN させて，文字を打ち込んでみましょうか。今 dfgsd…なんて最初の行に打ってみましたけど，ディスプレイには1文字ずつたてに表示されました。

```
run
d
f
g
s
d
j
g
s
d
j
g
d
Break in 10
Ok.
■
```

プログラムの10行を見て下さい。" "とありますが、これはスペースではありません。もしa$に何も入らなかったら（つまり、KEY入力がなかった時)、入力があるまで待ちなさいということですよ。キャリッジ・リターンを押さないで入力されてしまうので、例えばリアルタイムゲームの様に、スピードを要求される時便利です。

```
10 input a$
20 print a$
30 goto 10
run
? d
d
? f
f
? g
g
? s
s
? d
d
? j
j
?
Break in 10
Ok.
■
```

INKEY$の意味はわかったかしら？　この2つのプログラムを比べてみてください。INPUT 文では？と聞いてくれる点も違いますよね。INKEY$を使った例文を作ってみました。

```
list
10 a$=INKEY$:IF a$="" THEN 10
20 PRINT ASC(a$)
30 GOTO 10
Ok
■
```

これは，1文字をアスキーコードに変換するプログラムです。こんなプログラムには，INKEY$を使うのが適してますよね。

```
10 input "Try again (y or n)";a$
20 if a$="n"then end
30 goto 10
run
Try again (y or n)? n
Ok
■
```

```
list
10 PRINT "Try again (y or n)"
20 a$=INKEY$:IF a$=""THEN 20
30 IF a$="n" THEN END
40 GOTO 10
Ok
run
Try again (y or n)
Ok
■
```

これは，よくプログラムで使う判断文です。y，n 1文字で，進路が決まるわけで，この場合も INKEY$ は便利です。

(☞67)

CLEAR

```
a=1
Ok
print a
 1
Ok
clear
Ok
print a
 0
Ok
■
```

変数のみをすべて0にしてしまう命令です。プログラムを見てください。NEW と違うのはわかるでしょ？

REPEAT ON, REPEAT OFF

KEYボードを押している間，その文字を出しつづけてくれるのが，REPEAT ON。REPEAT OFF にしておけば，1回表示するだけです。(☞68)

67
INKEY$は1文字取り込みなさいという命令で、複数個の文字を取り込む時には使用できません。

68
REPEAT ON
リアルタイムゲームなどをPLAYする時、キーの先行入力が入ってしまい、肝心な時に思うような動作をしてくれなかったことはありませんか。そんな時にこの命令を使うと万事解決。先行入力は受け付けません。

REPEAT OFF
この**REPEAT OFF**を実行すると、プログラム作成中、同じキーを押し続けても1文字しか書いてくれませんよ。

```
repeat on
Ok
aaaaaaaaaa

repeat off
Ok
a■
```

AUTO

一行ごとに10番おきの番号を，自動的にふってくれる命令。AUTO 1,1 だったら 1 から 1 番おきに，AUTO 100,10だったら100から10番おきにうってくれます。こういう指定もできるんですネ！

```
auto
Ok
10 ■

auto 100,10
Ok
100 print "Yuri"
110 print "Shinagawa"
120 print "Hudson"
130 ■
```

DELETE

部分的にプログラムを消す命令。DELETE 30 ならば行番号30を消してくれます。DELETE 30～60 ならば30番から60番が消えますよ。

もちろん，30 RETURNとしても30番は消えます。ただ，30番から60番まで一度に消す事はできませんよね。RETURN を使うなら，30 RETURN，40 RETURN，50 RETURN としなきゃネ！

```
list
10 PRINT "Yuri"
20 PRINT "Shinagawa"
30 PRINT "Hudson"
40 PRINT "Soft"
50 PRINT "Dr.Bee"
60 PRINT "X1"
70 PRINT "SHARP"
Ok
delete 30-60
Ok
```

```
list
10 PRINT "Yuri"
20 PRINT "Shinagawa"
70 PRINT "SHARP"
Ok
█
```

EDIT

指定の行をリストしてくれます。1行だけを書き直したい時なんかに使いますね。

```
list
10 PRINT "yuri shinagawa"
20 GOTO 10
Ok
edit 10
10 █RINT "yuri shinagawa"
```

CONSOLE

実行する画面の指定です。(☞69)

つまり，y 軸の10から8行分とり，x の10から8行分とって，その範囲を画面としてしまうという事です。命令はその範囲の中で実行されるんです。

画面は，こんな小さく指定されてしまいましたネ！(図7)

図7

```
console 0,25,0,40
Ok
█
```

元の画面にもどす時はこの様に指定します。

```
list
10 PRINT "Yuri ";
20 GOTO 10
Ok.
console 10,8,10,8
Ok.
i Yuri Y
uri Yuri
 Yuri Yu
ri
Break in
 10
Ok.
█
```

TR ON

トレース機能を ON とする命令，といってもわからないでし

69

CONSOLE命令は、画面内の文字を表示するエリアを設定する命令です。

CONSOLE Y1,Y2,X1,X2

Y_1＝垂直方向の表示を開始する行位置（0～24）

Y_2＝垂直方向の表示行数（1～25）

X_1＝水平方向の表示開始文字位置
WIDTH40のとき　0～39
WIDTH80のとき　0～79

X_2＝水平方向の表示文字数
WIDTH40のとき　1～40
WIDTH80のとき　1～80

テキスト画面（文字のみを表示するモード）に対して、文字の表示エリアを設定します。この命令の実行後は、指定した長方形のエリア内だけに、文字を表示することができます。

画面クリアや、スクロールもこのエリア内で行われます。カーソルもこの設定位置内だけで移動します。

ょ。トレース機能とは，今実行している命令の行番号を頭に表示してくれる命令なんですね。

例文を見てみましょうか。

```
list
10 PRINT "Yuri ";
20 GOTO 10
Ok
tron
Ok
run
[10]Yuri [20][10]Yuri [20][10]Yuri [20][
10]Yuri [20][10]Yuri [20][10]Yuri [20][1
0]Yuri [20][10]Yuri [20][10]Yuri [20][10
]Yuri [20][10]Yuri [20][10]Yuri [20][10]
Yuri [20][10]
Break in 10
Ok.
■
```

TR OFF

TR ONにして始まったトレース機能を止めるのがTR OFF。この命令によって，またもとのプログラムを実行してくれるようになります。

```
list
10 PRINT "Yuri ";
20 GOTO 10
Ok
troff
Ok
run
Yuri Yuri Yuri Yuri Yuri Yuri Yuri Yuri
Yuri Yuri Yuri Yuri Yuri Yuri Yuri Yuri
Yuri Yuri Yuri Yuri Yuri Yuri Yuri Yuri
Break in 10
Ok.
■
```

WIDTH

画面のモードを指定する命令です。X 1は40文字モードと80文字モードの2つのモードをもっていましたよね。通常の状態では40文字モードですから，80文字モードにしたい時，または80文字モードから40文字モードに戻したい時に使いますね。

下のように字の大きさは，ぜんぜん違っちゃいます。

```
width 40
Hudson soft
width 80
Hudson soft
```

LABEL

ON…GOTO 文や，IF…GOTO 文などのように，条件によってジャンプさせる時に，とても有効な命令です。

ON…GOTO などでは，飛ぶ先を直接番号で指定してやらねばなりませんが，LABEL を使えば，GOTO の後に書かれたストリングと同じものを見つけて，そこへ飛んでくれるんです。

LABEL の役割は，ストリングのある場所を示す事なんですね。名前の通り，プログラム中にラベルをつけてくれるわけです。

```
10 a$=inkey$:if a$="" then 10
20 if a$="e"then "Yuri"
30 print a$;
40 goto 10
50 label "Yuri"
60 print "End"
run
abcdEnd
Ok
█
```

例えば次のプログラムでは，LABEL のついた文は実際には何もしませんが，飛ぶ位置を指定してプログラムの流れを決めてくれてますよね。(☞70)

```
LIST
10 a=INT(3*RND(1)+1)
20 ON a GOTO"Yuri","Shinagawa","Hudson"
30 GOTO 10
40 LABEL"Yuri"
50 PRINT "1";
60 GOTO 10
70 LABEL"Shinagawa"
80 PRINT "2";
90 GOTO 10
100 LABEL "Hudson"
110 PRINT "3";
120 GOTO 10
Ok
█
```

70

HuBASICならではの命令、LABEL（ラベル文）です。

プログラムを組んで行く上で、今後サブルーチンにしよう、この処理はあとで考えよう、などと頭の中で流れを考えて組んで行く時、サブルーチン等の飛び先が決まらない時に威力を発揮します。

```
GOSUB "ｹｲｻﾝ"
ON X GOTO "ﾀｼｻﾞﾝ","ﾋｷｻﾞﾝ","ｶｹｻﾞﾝ","ﾜﾘｻﾞﾝ"
```

この様に書いておき、処理ルーチンが決まったら、

```
LABEL "ｹｲｻﾝ"
LABEL "ﾀｼｻﾞﾝ"
```

などと、飛び先の頭にLABEL文を書いておくだけで良いのです。

もちろんリナンバーされてもだいじょうぶですよ。

サブルーチンを表示するREM文の代わりにも使えますね。

```
list
10 a=INT(3*RND(1)+1)
20 ON a GOTO 50,80,110
30 GOTO 10
50 PRINT "1";
60 GOTO 10
80 PRINT "2";
90 GOTO 10
110 PRINT "3";
120 GOTO 10
Ok
■
```

STOP と END の違い

さて，プログラムを見比べてみましょう。簡単に言ってしまえば，CONT（コンティニュー）ができるのがSTOPで，プログラムが完全に終わってしまうのがENDです。

ENDで終わったプログラムを続けようとしてもCan't continueという表示が出てきてしまいます。どちらの場合も，プログラムを一通り実行して止まるのは同じことです。

```
10 print "Hudson"
20 stop
30 goto 10
run
Hudson
Break in 20
Ok.
cont
Hudson
Break in 20
Ok.
■
```

```
10 print "Hudson"
20 end
30 goto 10
run
Hudson
Ok
cont
Can't continue
Ok
■
```

CFLASH 1

白と黒が，交互に点滅する命令です。この命令を入れておけば，プリント内容がチカチカと目立ちますから，強調させたい時にはいいですね。

CFLASH 0

フラッシュを止める命令です。

71

CREV　キャラクターリバース

この命令は文字の表示モードを、正常モードか反転モードかの切り換えを行う命令です。

CREV 0　ノーマルモード 正常モード

CREV 1　リバースモード 反転モード

この命令の実行以降、文字はすべて指定された表示モードで画面に表示されます。

ノーマルモード　ABC 123
反転モード　ABC 123

CGEN　キャラクタージェネレータ

キャラクタージェネレータを、標準の（内蔵のROM）を使うか、ユーザーキャラジェネ（自分で設定したキャラクタージェネレータ）を使うかの表示モード切り換えです。

CGEN 0　ノーマルモード (ROMキャラジェネ)

CGEN 1　ユーザー文字モード (RAMキャラジェネ)

このキャラクターの定義は、別の項で説明いたします。

```
cflash 1
Ok
cflash 0
Ok
█
```

CREV

CGEN

とってもむずかしくてわかりませんでした。もっと勉強しておきます。(☞71)

WHILE/WEND

WHILE と WEND は対にして使われ，この間でループします（同じことをくり返します)。ループする条件は，WHILE の後に記述し，条件が成立している間はループし続けます。わかるかしら…？　注意するのは，条件がループの最初で判断されている点で，場合によっては，一度もループしないで通りぬけることもあり得るんです。この点が，後であげる REPEAT～UNTIL との相違点です。

フローチャートにしてみましょうか……。(図8)

繰り返す操作の前に条件が判断されるんです。もう大丈夫でしょ！

図8

REPEAT/UNTIL

WHILE/WENDと考え方が似ているところもあるので，混同しないように！(図9)

REPEAT と UNTIL は対にして使用し，この間でループをくり返します。ループ条件は UNTIL の後に記述し，条件が成立するまでループを続けます。注意するのは，条件がループの最後で判断されている点で，少なくとも1度はループされる結果となりますよね。この点が WHILE—WEND と区別されます。

このように，UNTIL で成立しなければ何回でも REPEAT します。しっかり頭に入れておきましょう。

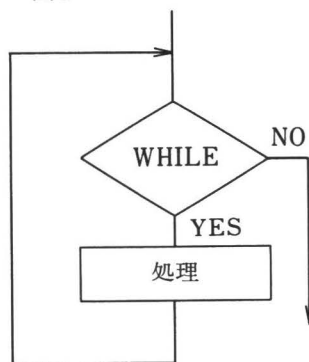
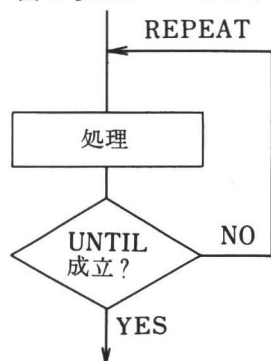



図9
このようにUNTILで成立しなければ、何回でもREPEATします。

DEF FN（デファインファンクション）

自分のよく使う関数をFNX(a)と定義する命令です。Xというのが関数式を示し，aは関数式に直接いれる変数です。

```
10 def fnx(a)=sqr(a)
20 input a
30 print fnx(a)

run
? 2
 1.4142136
OK
■
```

このプログラムが基本的な使い方です。

```
10 a=30
20 print fnx(a)
30 def fnx(a)=sin(a*pai(1)/180)

run
Undefined function in 20
Ok
■
```

DEF FNは，必ずはじめに指定してください。上のプログラムでわかると思いますが，先にFNX(a)を書かせる命令をしても，FNX(a)の内容を決めてあげていないわけですから，何も書けないでしょう？　もし，FNX(a)を，2度以上定義してしまったら，どんな事になるんでしょうか。

```
10 def fnx(a)=sqr(a)
20 def fnx(a)=sin(a)
30 a=30
40 print fnx(a)
run
Duplicate definition in 20
OK
■
```

私の予想では，新しく定義し直された方で実行してくれると思ったのに，結果は，この通り。一つの変数は，当然の事ですが一つの関数しか定義できないのです。

```
10 def fnx(a)=sin(a*pai(1)/180)
20 a=30
30 print fnx(a)
run
 .5
Ok
■
```

```
10 def fnx(a)=sin(a*pai(1)/180)
20 def fny(a)=cos(a*pai(1)/180)
30 a=30
40 print fnx(a),fny(a)
run
 .5            .8660254
Ok
■
```

これらは DEF FN を使った実行例ですよ。このように定義された関数が，長ければ長いほど DEF FN の利用価値がありますよね。今まで，式を通した結果だけを先に A$ などという形で定義しておく事は可能でしたが，DEF FN のように，式そのものの定義は無理でしたよね。実際にプログラムを書いてみると，この命令がいかに便利かがわかるようです。

```
10 for i=30 to 35
20 print sin(i*pai(1)/180)
30 next i
run
 .5
 .51503807
 .52991926
 .54463904
 .5591929
 .57357644
Ok
5 def fnx(a)=sin(a*pai(1)/180)
20 print fnx(i)
run
 .5
 .51503807
 .52991926
 .54463904
 .5591929
 .57357644
Ok
■
```

上の方のプログラムでは，30〜35度までの sin を求めているのですが，下の方のように DEF FN を使って書き直しました。

上と下は，結局同じことですが，上の20番の Print, sin (i＊pai(1)/180) は，今組まれている10番〜30番の FOR〜NEXT ループ内でしか実行されませんから，プログラムをのばして，もうひとつ FOR〜NEXT ループを入れ，同じ事をさせるなら，

また20番の式をどこかに入れてやらなければならないわけです。

DEF FNX(a) で1度定義した関数は，プログラムの中でどんなに飛んでも大丈夫！ FN X(a)とすれば，いつでも答えは出てきますよ。DEF FNX( ) の ( ) の中は，n 個。メモリのある限り入ります。次のプログラムはその例ですネ。

```
def fnx(a,b)=a*b+a/b
OK
print 2*3+2/3
 6.6666667
OK
print fnx(2,3)
 6.6666667
OK
█
```

```
10 def fnx(a,b)=a+b
20 a=2:b=3
30 print fnx(a,b)
run
 5
OK
█
```

```
def fnx(a,b,c,d,e)=a*b+c*d/e
Ok
█
```

DEFSTR（デフストリング）

次はデフストリングという命令です。

```
def str a
Syntax error
Ok
defstr a
Ok
a$="Yuri"
Ok
a="Yuri"
Ok
print a
Yuri
Ok
a=123
Type mismatch
Ok
█
```

さっき勉強したDEF FNのように，DEF STRと離して書いたらエラーメッセージが出てきちゃいました。デフストリングはDEFSTRと書きます。

この命令はとくに難しいものではなくて，ストリングを変数で定義する時，$ マークを省いてよいというものです。a$=〝YURI〞を DEFSTR a としておけば，a=〝YURI〞と書いても良いのです。DEFSTR 後に指定した変数は，その後のプログラムではすべて $ がついていると考えてください。ただその $ が隠れていて，目には見えないというだけ……。

前のプログラムの中で a=1 2 3 と書いて，なぜタイプミスマッチになっちゃったかはわかったでしょ。A$=1 2 3 と書いているのと同じだからですよね。

```
defstr a,b,s,z
Ok
defstr a-z
Ok
z="Yuri"
Ok
y="Shinagawa"
Ok
a=z+y
Ok
print a
YuriShinagawa
Ok
█
```

DEFSTR a,b,s,z とすれば，a，b，s，z からはじまるものすべての後に $ が隠されていると思ってください。

```
defstr a
Ok
abcd="Media"
Ok
print abcd
Media
Ok
a1="Soft"
Ok
print a1
Soft
Ok
print a1;abcd
SoftMedia
Ok
█
```

もうわかったと思いますが，DEFSTR a としてあったとしたら，a=〝YURI〞でも，abcd=〝YURI〞でも良いという事です。DEFSTR a～z としておけば，頭の文字が a から z までだったら，

すべて＄がついていると考えられるわけで，どんなものでもそのあとに＄がついているのと同じ事だといえます。

```
a$=Yuri
Type mismatch
Ok
a=1234
Ok
defstr a
Ok
a=1234
Type mismatch
Ok
a="Yuri"
Ok
print a
Yuri
Ok
■
```

DEFINT（デフインテジャー）

これもこのようにつなげて書いてください。ある程度予想がつくでしょうが，この指定を入れたあとに出てきた数字は，全部小数点以下を切り捨てた整数の形で表示されます。プログラムを見れば，一目瞭然！

```
a=3.1415927
Ok
print a
 3.1415927
Ok
defint a
Ok
a=3.1415927
Ok
print a
 3
Ok
■
```

```
defint a-z
Ok
defint a,f,z
Ok
■
```

指定の仕方などは，すべて DEFSTR と同じ。こんな命令があれば，サイコロゲームなんて簡単ネ。どうしてもっとはやく教えてくれなかったんでしょうネ……。

DEFINT の応用例として，サイコロゲームのプログラムをあげておきましょうね。

```
LIST
10 DEFINT a
20 a=RND(1)*6+1
30 PRINT a
Ok
run
 4
Ok
■
```

DEFDBL（デフダブル）

この指定をしておけば，全て倍精度（16けた）で表示してくれます。下に例がありますからわかるでしょ。通常は単精度ですから，有効数字８けたまでで，８けた目は９けた目を四捨五入した数字となります。ここでは，１２３４５６７８９Ｅ+15となってますね。その他の指定の仕方などは，全て DEFSTR, DEFINT と同じですから参照してください。

```
a=1234567890123456
Ok
print a
 1.2345679E+15
Ok
defdbl a
Ok
a=1234567890123456
Ok
print a
 1234567890123456
Ok
■
```

DEFSNG（デフシングル）

DEF FN, DEFINT, DEFSTR 等，ディファイン（デフとは実は DEFINE;定義するの略だったのです）命令を解除するためのものです。DEFSNG a, b とすれば，a, b から始まるものだけが解除されますから，部分的にもとに戻すことも可能なわけですよね。

その他，DEFINT の指定をしてある時に，突然小数点以下第１位を調べたいという場合に，便利です。

```
defsng a-z
Ok
defsng a,b
Ok
■
```

NOTE7
コンピュータサウンド
パソコンでテクノサウンドを演奏してみる

コンピュータに音楽を演奏させることができるなんて，知っていました？

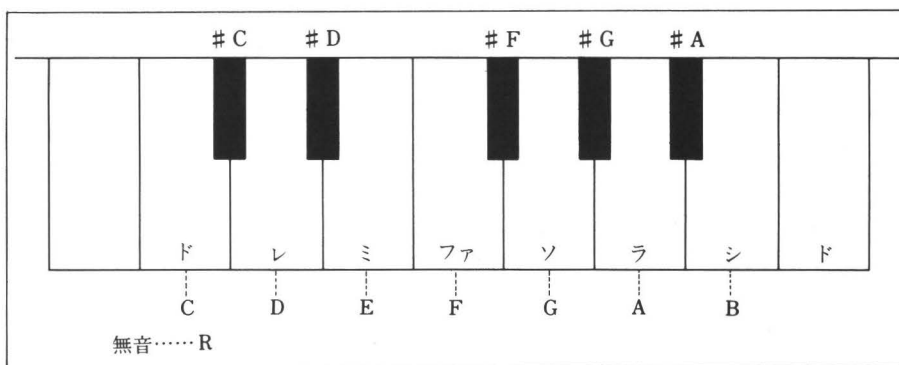
こんな機械にホントに音が出せるのかしらねえ……？　先生は「バカにしちゃいけませんよ」なんて言ってます。

まずPLAY（演奏）と音名（CDEFGAB）とそれにR（休符）の指定を憶えましょう！

```
PLAY"CDEFGAB"
Ok
▒
```

まあ，本当に音が出たわ！ PLAY が音楽を演奏しなさいという命令です。(☞72)

音名の指定は CDEFGAB というようにします。

この８音にプラスして，＃の記号も使えます。半音の指定は，全部このシャープでするんですって。

このようになりますよね。それから，休符の記号はRですって。たぶん英語の REST からきてるんじゃないかなあ……。

72

Ｘ１はプログラマブル・サウンドジェネレータ回路を持っており、各種の効果音を容易に作り出すことができます。内蔵スピーカー又はテレビのスピーカーから、８オクターブ３和音の音楽を楽しむことができます。

この項は、音楽家のユリちゃんの説明で進行いたしましょう。

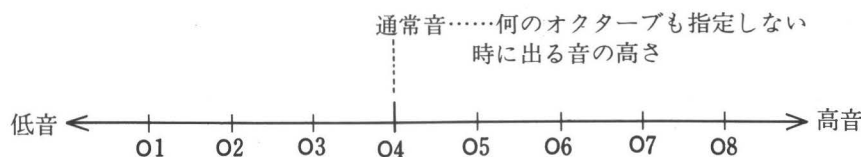
次にオクターブ【音の高さ】と長さの指定、これで音楽の要素が全て出揃ったワ!

さて，音名を指定したので，次はオクターブと音長を順番に説明していきましょう。

さっきの PLAY “CDEFGAB” は……

```
PLAY"O4CDEFGAB"
```

と同じ事なんです。

オクターブは8個までで，基準になっているのがO4。つまり，O4を中心に，数が大きい方が高音となります。

O1,…O8という記号を使って、オクターブはいつでも自由に変えることができるの

いいかしら？　高さは一度指定しておくと新たに指定してやるまではその高さを持続しているみたい。

```
PLAY"O4CO5CO3CO5C"
```

ゼロじゃなくてオーですヨ
O4が基準になります

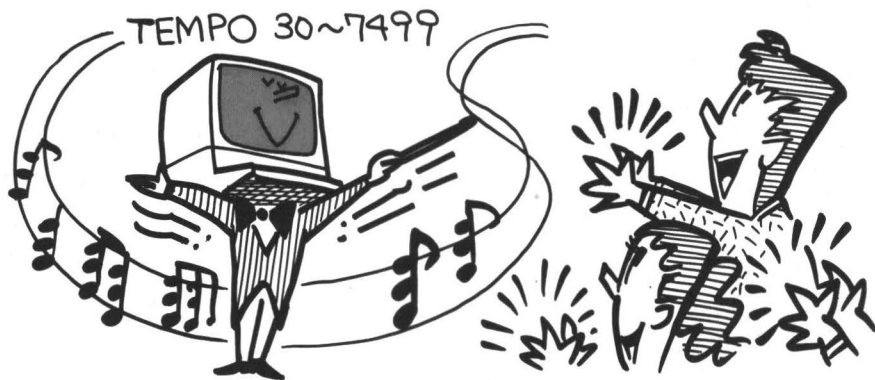
こんなかんじで，一音ごとにオクターブをかえたりもできちゃいます。

```
PLAY"O3A"
```

この音の波長は440 Hz（ヘルツ）になるそうです。オーケストラが音を合わせる時なんかに使う音なんですって。ギターならば第5弦，バイオリンなら下から2番目の弦の音だそうで，みんなこの音にあわせるみたいです。

音長の指定は32分音符𝅘𝅥𝅰から全音符𝅝までを0〜9までの数字で表現しましょう!

高さの指定の次は，音の長さを指定します。32分音符から全音符までを，0から9までの数字で指定するんですネ（休符Rをつけると休符の長さになります)。

同じ音長の音符が続くときは，2番目の音符からは音長の指定は省略できます。はじめから音長が指定されないと，4分音符（5の音長）とみなして実行してくれます。

曲の速さはTEMPO命令で入力、30~7499ステップまで指定できるんですヨ!

音長の指定はやりましたけど，今度は曲全体を早めたり，ゆっくりしたりできる命令。**TEMPO X**（X=30~7499）です。

TEMPOは120が基準なので，何の指定もしなければこのテンポで演奏します。

数の大きさに比例して，テンポははやくなります。7499なんて，どんな速さかしら？

```
10 TEMPO 7499
20 PLAY "CDEFGAB"
30 GOTO 20
```

えっ?! あまりの速さに，しばしあ然。でも，どこかで聞き覚えのある音ねえ……。そうだ! よくゲームセンターで聞くような音ですね。

和音は音名を:コロンでつないで作るのです。ただし3音までが限界なのです

さて，最後にもう1つ便利なことがあるんです。これで，ナンと和音がつくれちゃうんですって。ただし，3音というか，3つのパートまでという制限はあります。

```
PLAY"C5:E5:G5"
```

今，ドミソの和音が聞こえてますよね。これは，こんな形でコロンでつなげば OK!

……と，まあこんな使い方もできますが，一番実用的なのは，次のようなプログラムだと思いますよ。

```
PLAY"G2G4G2G4G2:F2E4E2E4E2:O2R2C2E2G2C2E
2G2"
```

これはバイエル90番の初めの部分です。

この曲のように，3つまで（もちろん右手の部分の和音だけPLAY することだってできます）のパートにわかれた曲を，一小節ぐらいずつ PLAY させるのにとても便利でしょ。

あとから見る時わかりやすいように，一つのストリングスは1小節，と決めておいた方が楽でしょうね。

ソプラノ、アルト、バスの3部演奏で“HOME SWEET HOME”を実際に演奏させましょう!

さて，では何か知っている曲をコンピュータに演奏させちゃいましょう。

```
ﾊﾆｭｳﾉﾔﾄﾞ HOME SWEET HOME

LIST
10 TEMPO 100
20 PLAY"O6C5E6F2R0F6G3:O4R5C1EGO5C1EGO6C
1O5GO6D1O5B1GFDO4B1GF"
30 PLAY"O6G7E5G5:O4C1EGO5C1EGO6C1O5GGE#C
1O4AGE#C1O3A1"
40 PLAY"O6F6E3F5D5:O3A1O4D1FAO5DFAO6D1BG
FDO5B1GFD"
50 PLAY"O6E8C3D3:O3C1EGO4C1EGO5C1EGECO4G
1ECO3G1E"
60 PLAY"O6E6F2R0F6G3:O4C1EGO5C1EGO6C1O5G
O6D1O5B1GFDO4B1GF"
70 PLAY"O6G7E5G5:O4C1EGO5C1EGO6C1O5GGE#C
1O4AGE#C1O3A1"
80 PLAY"O6F6E3F5D5:O3A1O4D1FAO5DFAO6D1BG
FDO5B1GFD"
90 PLAY"O6C8R5:O3C1EGO4C1EGO5C1EGECO4G1E
CO3G1E"
Ok
```

これはソプラノ，アルトの2部演奏ですヨ。

効果音の合成はSOUND命令と2つの8進数&O17なんかの組み合わせで処理!

これで，音楽関係の命令はおしまいかな？　いいえ，まだあるのです。

```
LIST
10 REM *** GUN SHOT ***
20 SOUND 6,&O7
30 SOUND &O7,&O7
40 SOUND &O10,&O20
50 SOUND &O11,&O20
60 SOUND &O12,&O20
70 SOUND &O14,&O70
80 SOUND &O15,0
Ok
▒
```

バーンという GUN SHOT の音が聴こえたでしょ。**SOUND** という命令をつかって，いろいろな音がつくれるみたいですネ。

例えば SL 機関車のシュシュという音とか，パトカーのピーポーピーポーという音とかね。

このプログラムは，実はとても難かしいんだけど，なるべくわかりやすく説明しましょうね。

&O17 とか &O7 とか，これは一体何かと言うと，8進数の表示なんです。&O7 は10進法の7，& O17 は15です。1の位は7までで8になってひとけたあがるという，あの考え方。2進数も出てきているから分かりますよね。

結局，SOUND のあとに数字を2つ指定してあるわけです。最初がアドレス，もう1つがデータとなるそうなんですが，この2つを指定しないと，コンピュータから音は出てきません。

& O17 そのものが何を意味するかというと HuBASIC の基の機械語というのが分からないといけないんですって。音を作るのってけっこう大変なんですよ。

NOTE 8
グラフィック処理に挑戦!!
BASICを学ぶ上で最も楽しいコンピュータ・グラフィックスの世界

HuBASICの特徴の中でも，特にメリットが大きいのが**グラフィック命令**なんです。(☞73)

簡単なプログラムで，多彩な表示が実現できたり，色の変更を瞬時に行うことができたりと，とてもおもしろそう。まず，コマンド(命令のこと）をひとつずつ見て行きましょう。

0～7というように、番号を各色に対応させるのがPALET命令なのです

グラフィックでは，大変重要な命令らしい **PALET** からいきましょう。一体どんな命令なのかな？

0	1	2	3	4	5	6	7
ブラック	ブルー	レッド	マゼンタ	グリーン	シアン	イエロー	ホワイト

（P.189のカラー写真参照）

まず，0〜7まで番号をふった7枚のスライドがあると考えてください。0番は黒，1番は青というように，番号により色がついています。今，7枚のスライドに同じように光が投影されているとしましょう。

4番のスライドはグリーンだから、4番の命令を与えてやると、図形は緑色に…

PALET 4

たとえば，こんな命令をしたとすると，4番のスライドを選んだことになります。4番はグリーンですから，何か描かせる命令（あとでやりますからね！）を与えてやれば，すべてグリーンで描いてくれるわけです。(☞74)

同様に，PALET 1ならばブルーで，PALET 6ならばイエローで描いてくれる，というのはわかるでしょ。では，こんな命令だったらどうするんでしょうか。

はじめに6番のイエローを入れておいて、あとで1番のブルーに入れ換えも自由

PALET 6，1

困ったなあ。何色（なにいろ）になるのかしら？　さて，ここがこの命令の一番肝心なところ！　さっきのスライドの説明を思い出してくださいネ。

スライドは，さし換えも可能です。この場合，最初は6番，つまりイエローのスライドを入れておいたのを，後で1番のブルーと入れ換えてしまったと考えればいいんです。

結果としては，今までイエローで描かれていたものが，すべてブルーになっちゃうわけです。

これがカラーパレットを変更するという事。大変便利なんで

74

PALET　パレット

だれもがパレットという言葉を使ったことがあると思いますが、いかがでしょうか。そうです。水彩画を描く時に使うあのパレットなのです。絵の具を混ぜ合わせて色をつくる板のことですね。

しかし、光には赤、緑、青の三つの絵の具しか無いのです。これを光の3原色と言うのですが、この三つの色を混ぜ合わせて色々な色を作り出さなければなりません。

X1にはこの三つのグラフィックメモリーがあります。

グラフィック2 R 赤　　グラフィック3 G 緑

この混ぜ合わせを指定する命令がパレット命令なのです。

PALET 0〜7，0〜7
パレットコード　カラーコード

PALET コード			COLOR コード		
0	黒	ブラック	0	黒	ブラック
1	青	ブルー	1	青	ブルー
2	赤	レッド	2	赤	レッド
3	紫	マゼンダ	3	紫	マゼンダ
4	緑	グリーン	4	緑	グリーン
5	水色	シアン	5	水色	シアン
6	黄	イエロー	6	黄	イエロー
7	白	ホワイト	7	白	ホワイト

パレットの各色に、カラーをそれぞれ混ぜ合わせて色を構成していくのです。

COLOR 0〜7，0〜7
表示色　背景色

文字の色と背景の色を指定する命令です。電源投入時は、自動的にCOLOR7，0（文字は白、背景は黒）にセットされています。グラフィック画面において、パレットコードを省略すると、COLOR文の表示色のコードの値が、パレットコードの値として使用されます。

グラフィックモードを使用する時は、PALETのみ指定し、パレットコード、カラーコードを省略すると、COLOR命令の色のみで実行してくれます。

すよ。

今まではブルーで描いた図を，色だけレッドに変えるなんて事はできなかったから，いちいち命令を書き直してたんですって。偉大な命令でしょ。

どちらの色が，どちらの色にとって換わるかは，間違えないように。右に書かれた番号の色の方が影響力があるわけなので，6番は，全部1番の色にかわってしまうんですヨ。

カ　ラ　ー	B（ブルー）	R（レッド）	G（グリーン）
0　ブラック	0	0	0
1　ブルー	1	0	0
2　レッド	0	1	0
3　マゼンダ	1	1	0
4　グリーン	0	0	1
5　シアン	1	0	1
6　イエロー	0	1	1
7　ホワイト	1	1	1

0とか1がごちゃごちゃ並んでますが，この表は何でしょう。

この表は，0〜7番までの色の構成を示しています。各色は，**R**(Red) **G**(Green) **B**(Blue)の3色を，いかに使うかにより構成されているわけなんですネ。この3色が**光の3原色**です。

表の見方なんですが，その色を使っている場合は1，使ってない場合は0という2進法で表わしてあります。では，ちょっと実際に見てみましょうか。

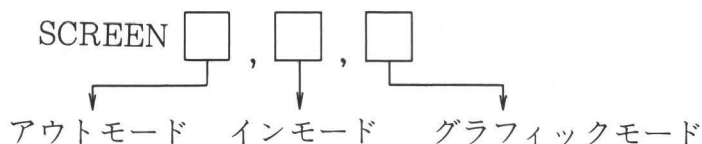
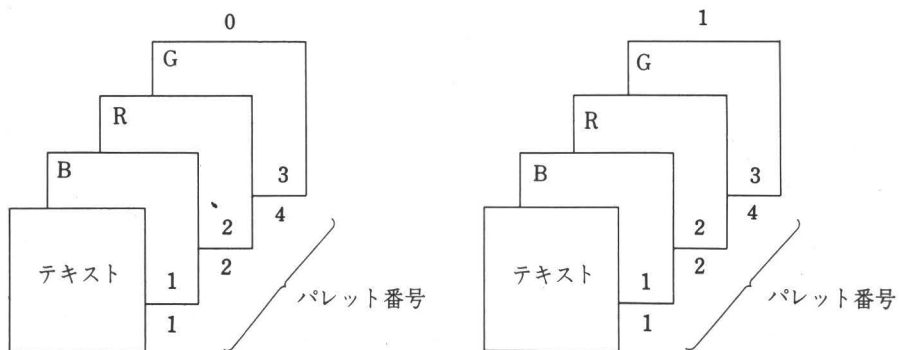
4番のGreenは，Gが1で，B，Rは0でしょ。

5番のシアンはどんな色かしら。青と緑が1だから，青緑といったかんじの色なんでしょうね。

0番のBLACKや7番のWHITEというと……。表のように，全て0の状態で黒，全て1の状態で白です。光の3原色をまぜると白になるっていうのを習った事があるでしょう。

グラフィックに使われている色は，こんな構成になっている事も頭のすみに入れておいてくださいネ。

スクリーン図を想定して、インやアウトのモードを使い分けるのがSCREEN命令

次は**スクリーン命令**です。これも PALET に劣らず重要。要するにスクリーンを指定する命令なんですが，指定の方法は？(☞75)

こんな形をとるそうです。でも，これじゃ何の事やらさっぱりわからないわよねえ。

さて，この図がこれからの POINT になりますからネ！ これを **SCREEN 図**と名づけておきましょう。

75

SCREEN　スクリーン命令

SCREEN 出力ページ，入力ページ，グラフィックモード

出力ページ……0か1（0の時ページ0，1の時ページ1を指定します）

入力ページ……0か1（0の時ページ0，1の時ページ1を指定します）

グラフィックモード…0，1，2，3

0…指定されたパレット番号で実行されます。

1…グラフィック1(青)のみ実行されます。

2…グラフィック2(赤)のみ実行されます。

3…グラフィック3(緑)のみ実行されます。

出力・入力のページとは、40文字モードの時のみ有効で、0と1の2ページのうち、どちらを表示して、どちらに書き込むか指定するものです。

☆**入力ページ**とは、画面制御命令やPRINT、INPUT命令等が実行されるページです。

☆**出力ページ**とは、実際画面に出力されているページのことをいいます。

☆**グラフィックモード**とは、3枚のグラフィック画面の使用モードです。

アウト・モードは実際に見える画面を指定するMODEで、0ページと1ページがあるの

アウト・モードは，実際に目に見える画面を指定する部分です。画面には，0ページと1ページの2枚があると考えてくださいね。SCREEN図の上の方に書いてある0と1がページ数です。

ディスプレイに現われる画面は，1枚分だけですから，アウト・モードの図の中に0か1のどちらかを入れて，ページを指定する必要があるわけですネ。

これは，目に見える（ディスプレイに現われる）画面の指定であって，何か書き込む画面とは，また違いますから気をつけて。

ただ，ここまでの話はすべて40文字モードでの事。モードについては覚えてるでしょ？　**WIDTH80**とすると，モードが80文字に切り換わって，違ってきちゃうんですよ。

80文字モードという事は，1画面に**40文字モード**の倍の文字が入るという事でしょ。だから，0ページと1ページに分けて書かれていた内容は，0ページ目にすべて収まっちゃう事になりますよね。

ページが2枚使えるのは40文字モードの場合で，80文字モードになると，アウト・モードにしろ，イン・モードにしろ1枚目。つまり0ページしかないのだという事です。わかったかしら……。

イン・モードは書き込む画面の指定。アウトの画面と番号を合わせると表示が見えるの

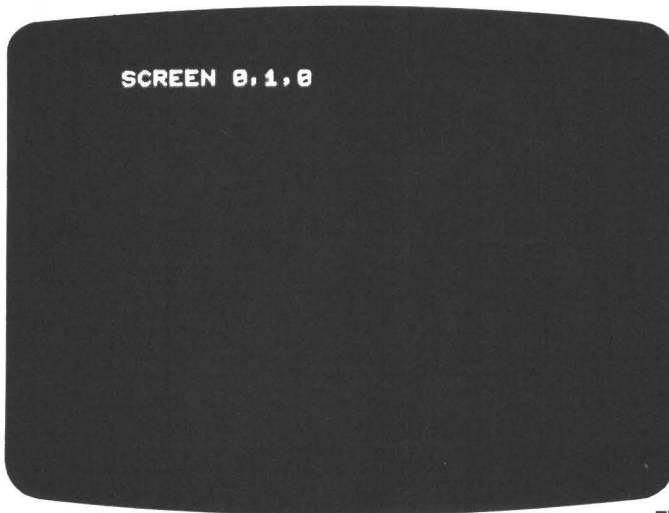
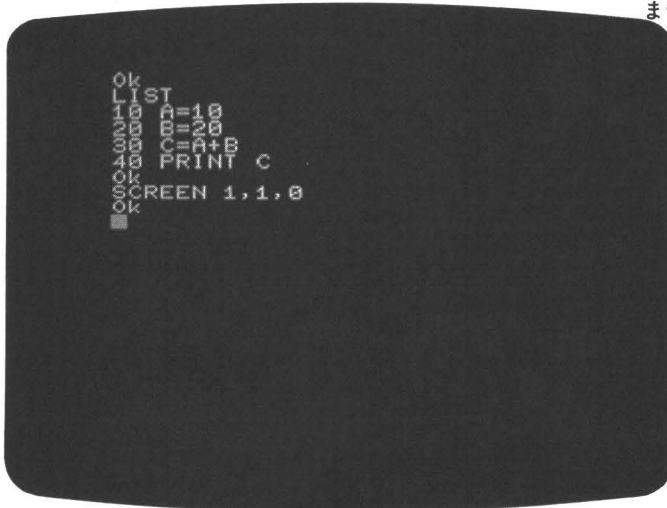
次に**イン・モード**は，何か書き込む面の指定になります。今，

ディスプレイには0ページが出てるとしますよね。この時，当然アウト・モードは0になっています。

ここで，イン・モードも0にしてやれば，プリントする内容はそのまま表示され目に見えますよね。

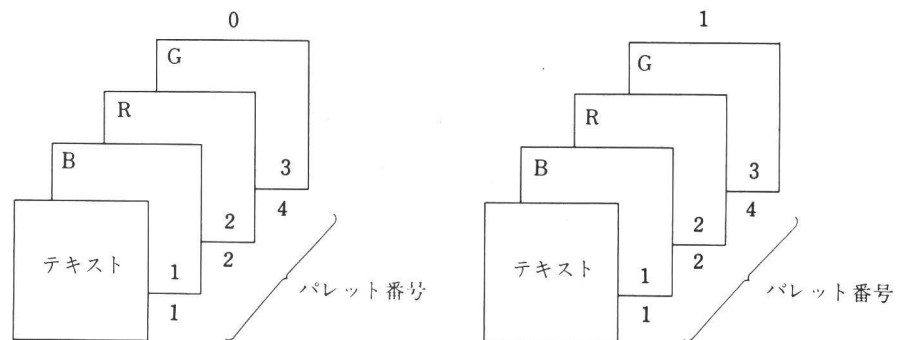
ところが，目に見えてない画面，つまり1ページ目に書き込んでいくことも可能なんです。

ディスプレイには出てこなくても，1ページ目にはきちんとプログラムが打ちこまれています。

さあ，アウト・モードを1にしてみてください。ディスプレイには，プログラムが現われますよ。今説明したところを，グラフィックモードを無視して考えると……。

ディスプレイには現われませんが、こう打っているのです。

（P.189のカラー写真参照）

ほらね！

グラフィック・モードは、対応する色番号を指定すると、全てその色で図形が出現

3つめの指定は，**グラフィック・モード**です。この3つのモードがわかってないと，**SCREEN 命令**は完璧とは言えないのです。さあ，またあの SCREEN 図へ戻りましょう。

今度は，この **TEXT，B，R，G** の4面が問題になってきます。

0ページ目，1ページ目両ページについて，この4枚の画面が使えると考えてください。

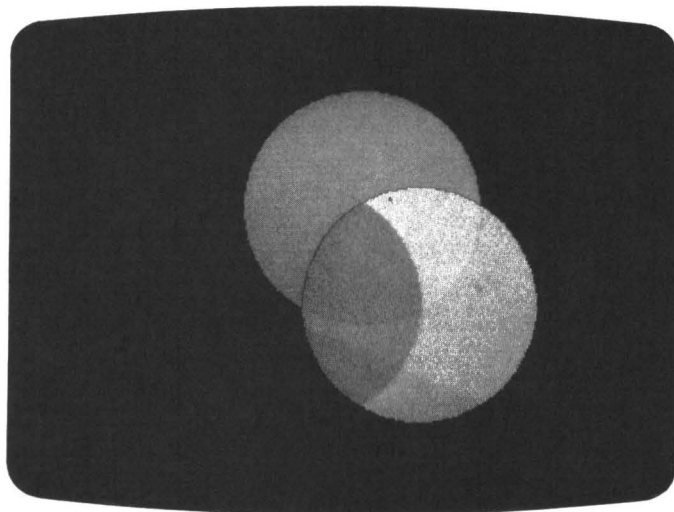
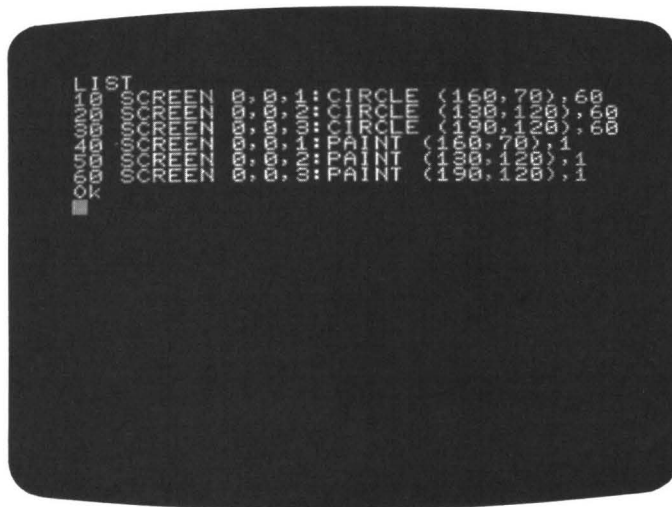
まず，ふつうに文字をプリントする場合は，**TEXT 画面**が使われます。この場合は，グラフィック・モードは関係がないので，特に指定はいりません。グラフィック・モードの図の中に，1と入れてみましょうね。1はBの画面なのです。

BはBlueのB。もう想像はつくでしょうが，1と指定しておくと，グラフィック命令の実行は全部青で行われます。

同様に2はRed，3はGreenである事は，わかるでしょ。

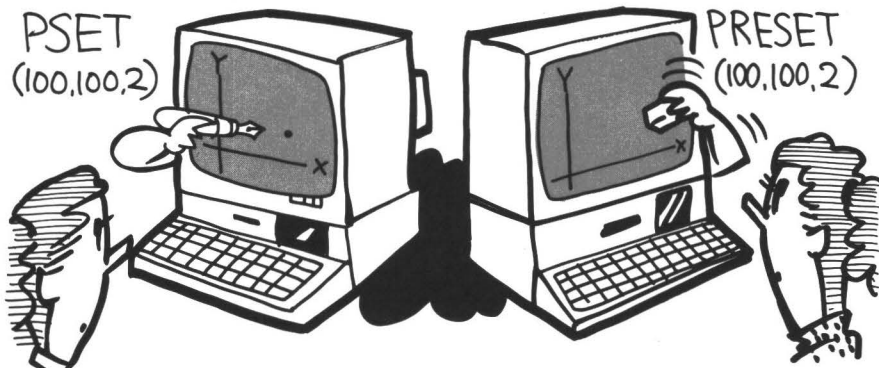
この場合，ひとつの画面で行われるグラフィックはすべてその色になるわけで，他の色の影響は一切うけません。

黄色で線をひきなさいという命令を与えても，グラフィック・モードが1ならば，全部Blueとなってしまうという事です。

以上3つのモードで，構成された命令がスクリーンです。スクリーン指定は，グラフィックではいつでも重要なんですョ。

（P.189のカラー写真参照）

B, R, Gとも他の色に影響を与えないので，こういうふうに出てくるわけです。

CLS0はRGB全ての画面を消せという命令で、CLS1～CLS3では、どれかの色が消去

次に**CLS0**って何だかわかるかしら？ これは，B, R, Gすべての画面を消せという命令です。CLSだとテキスト画面のみが消え，CLS1～CLS3 はそれぞれ B,R,Gの画面のみが消えるわけ。CLS 4 は何だと思いますか？これはテキスト画面とB, R, Gの全てを消しちゃうんですね。

PSETは(X,Y)の座標にC色で点を打ちなさいという命令でPALETと併用ネ!

さて，それでは PSET に移ります。

これは別に難しい命令じゃなくって，“(X, Y) の座標にC色で点をうちなさい” という命令なんですって。

つまり，X, Y はそのまま X 座標，Y 座標の位置であり，C は PALET 命令の Color Number (0〜7) がそのままあてはまるわけよね。ちょっとやってみようか!! (☞76)

(P.189のカラー写真参照)

なる程。(100, 100) のところに小さな赤い点がうたれました。こんな風になるのねえ!

76

ドットのコントロール

PSET　PRESET

ポイントセット　ポイントリセット
グラフィック画面にドット(点)を描いたり、消したりする命令です。

PSET(x, y, p)

水平位置, 垂直位置, パレットコード

パレットコードを省略すると、その前にCOLORコマンドで指定した表示色になります。

グラフィック画面の座標X，Yの点を、指定のパレット番号でセットします。マルチ画面モードの時は、指定の画面のみ、パレットコード0ならリセット、それ以外ならセットします。

PRESET(x, y, p)

PSETと同様指定された点を、指定のパレットコードで消します。

PRESETはPSETと対に使って、指定した点を消す命令なの。ただし指定の色で…

PSET は点を指定する命令でしたよね。PSET と対にして憶えなきゃならないのが，この **PRESET**。想像がつくんじゃないかな？

そう！これは指定した点を消す命令なのです。ただし，指定した色で消すという所が POINT。

例えば PSET (100, 100, 2) としたならば，必ず PRESET (100, 100, 2) としなきゃだめなのです。これを2のかわりに他の色番号にすると，おかしな事になっちゃうんですよ。

わかりやすい例でいくと，たとえば黄色で指定した点を，緑で消そうとしたとするでしょ。

そうすると，なんと赤い点になっちゃうワケ。このあたりの原理は，さっきの PALET とまったく同じなんですネ。

	青	赤	緑
6　黄	0	1	1
4　緑	0	0	1
2　赤	0	1	0

ここでは引き算になっているんです。分かるかしら？

このようにすればわかるでしょ？

つまり，6を4で消しても，4の色のみ消すわけだから，赤の部分の1は依然としてそのままであり，結果として赤になっちゃうわけです。おもしろいでしょ。

まあ，点の色を変えたいなどという時には，今の考え方を応用すればいいわけよネ。

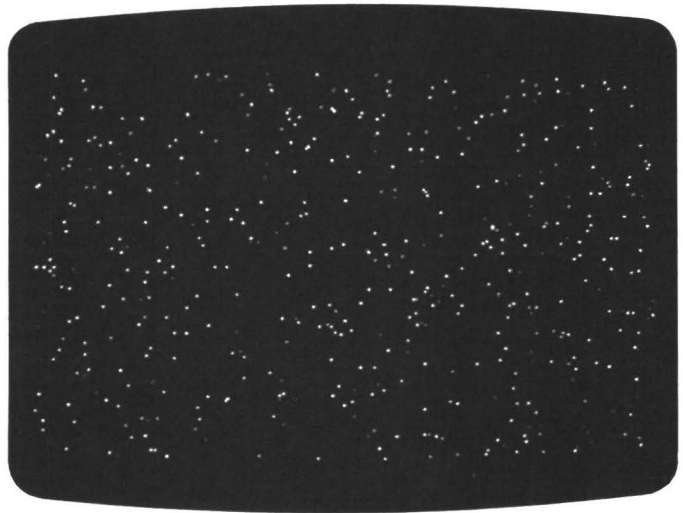
さて，ではこの辺で，グラフィックのプログラムを書いてみたいのですが，大丈夫かしら？　まあ，先生もついている事だし，挑戦してみましょう。

点をRNDに打つと、画面いっぱいに7色の点がバラバラに1001個出現!

CLS0とは0画面を消すこと。
なぜ0画面を消したかというと、まだ20番以降にプログラムが続くからですよ

```
LIST
10 SCREEN 0,0,0:CLS 0:PALET
20 FOR I=0 TO 1000
30 X=RND(1)*640:Y=RND*200:C=INT(RND*7)+1
40 PSET (X,Y,C)
50 NEXT
Ok
```

0から6までですが、0は黒で画面と同色で見えないので1足すことで1〜7とします

さて，このプログラムを実行させれば，画面一杯に7色の点がめちゃくちゃに1001個うたれるはずなんですよね。どうかなあ？ちょっと心配……。

(P.189のカラー写真参照)

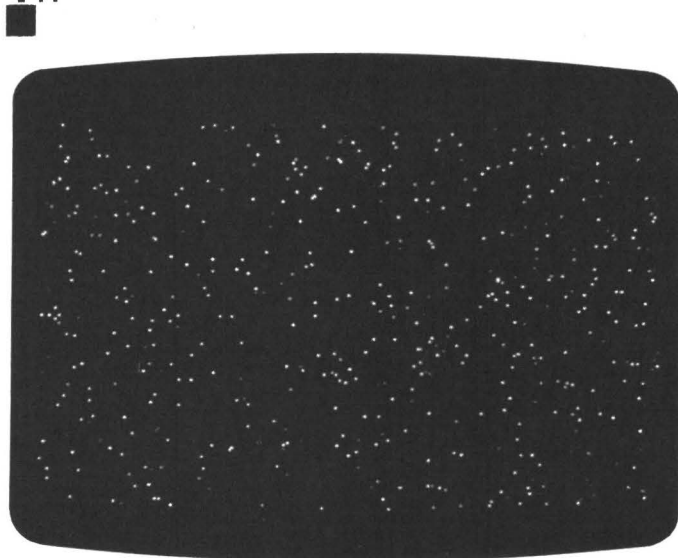
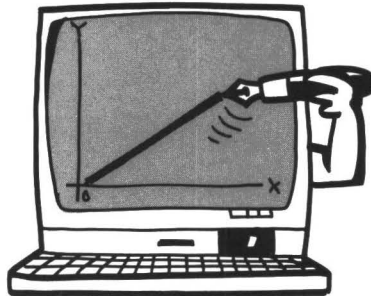
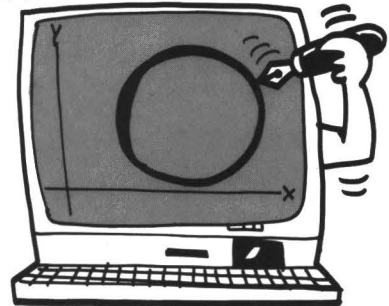
どうやら成功のようですよ。もう一回プログラムを見てみましょうか。

あれ？ LIST したら自然にグラフィック画面が消えたワ！

そうなんです。L. することで自動的に画面がきりかわるん

です。

じゃあ，再度グラフィック画面を見る時はどうしたらいいんでしょうか。

```
SC.0,0,0
Ok
■
```

こんな風に，再度命令を与えてやればいいわけですね。

LINEは"線を引く"という命令。引き始めと引き終わりの点を指定して…

LINE は，その名の通り“線を引く”という命令です。どのように指定するんでしょうか？（☞77）

このような指定，**モード**と **PAT(パターン)** については，ちょっと説明がいりますねえ。

モードについては，PSET（点をうつ）を知っていればほとんど用は足りるらしいのですが，この他には **XOR** と言って，2回書くともとに戻るモードなどがあるようです。

次はパターンですが，これはどんな LINE のパターンかと言うことで，通常の実線は＆HFFFF というパターンなんですって！

この場合は指定しても，あるいは何も指定しなくてもよいのだそうですよ。

それ以外のパターンとして，点線とか一点破線等，何でも書けちゃうんですよ！

77

画面に線を引いたり、箱を描いたりする時は、PSETで1点ずつ指定しても良いのですが、それは大変な作業になってしまいます。そんな時に、

LINE　命令で解決!!

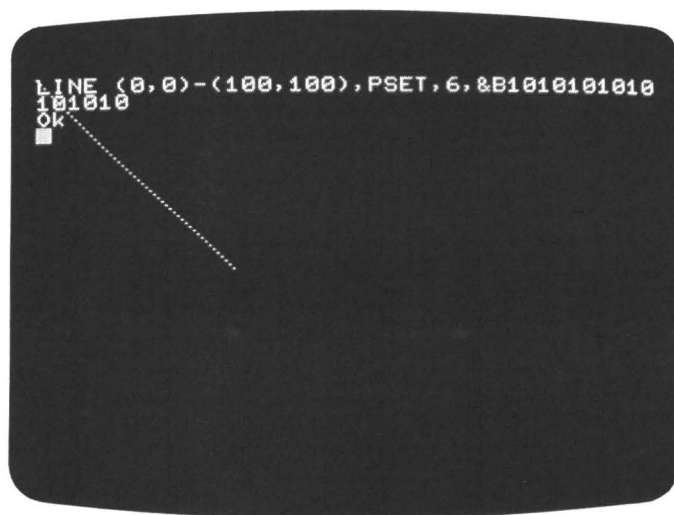
```
LINE(X1,Y1)-(X2,Y2),
PSET,C
```

x1とy1の位置を出発点として、x2とy2の位置を終点とした線を、Cで指定された色で引く。

```
LINE(X1,Y1)-(X2,Y2),
PSET,C,B
```

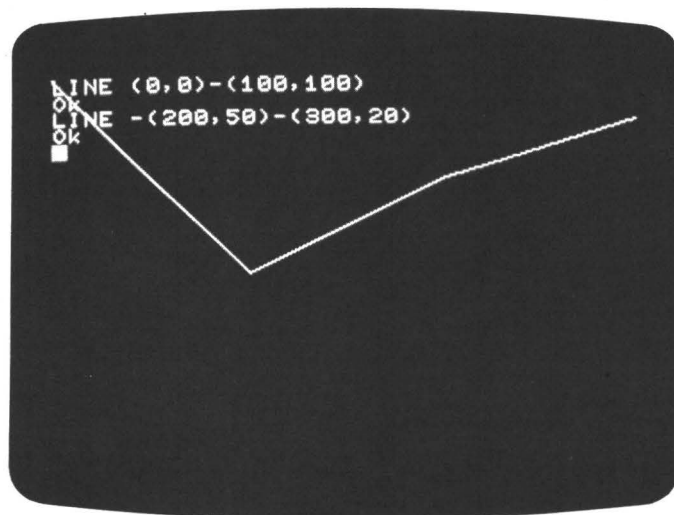
x1とy1の位置と、x2とy2の位置を対角線とした箱を、Cで指定された色で引く。

```
LINE(X1,Y1)-(X2,Y2),
PSET,C,BF
```

x1とy1の位置と、x2とy2の位置を対角線とした箱を、Cで指定された色でぬりつぶす。

(P.189のカラー写真参照)

これは（0, 0）から（100, 100）まで，黄色で1コマおきの点線がかけるんですよね。わかるかしら？

つまり，＆Bのあとの1010……というのが今の場合のパターンで，1の時は点を打ち，0の場合は打たないという事です。この101…が計16個（16ビット）書いてありますけど，必ずこのように，16ビットまで指定してください。

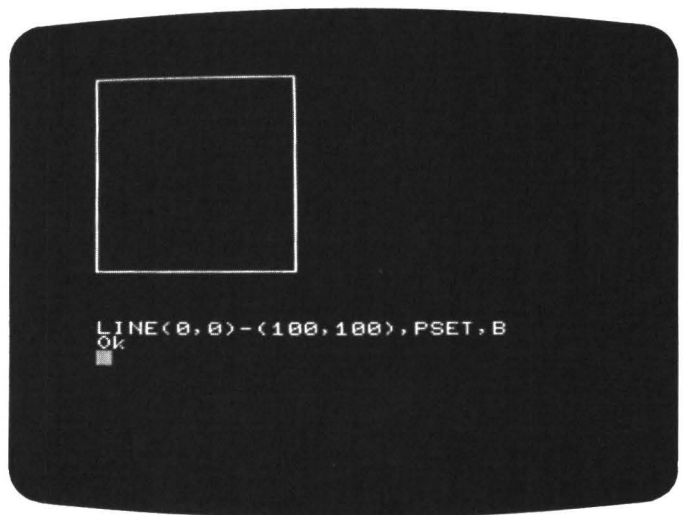
LINE命令では、このように複雑な線も描けます。色を変えてまだら線もできるの

(P.189のカラー写真参照)

さて，LINE 命令だとこんな風に線を連続させることも可能です。

色なんか変えてみて，ランダムに線をつなげていったらきれいなんじゃないかなあ……。いろいろ遊んでみてください。あっそうそう！キャラクターラインもひけるんです。

BOXは矩形を描く命令だけど、指定したラインを対角線とする長方形を作ります

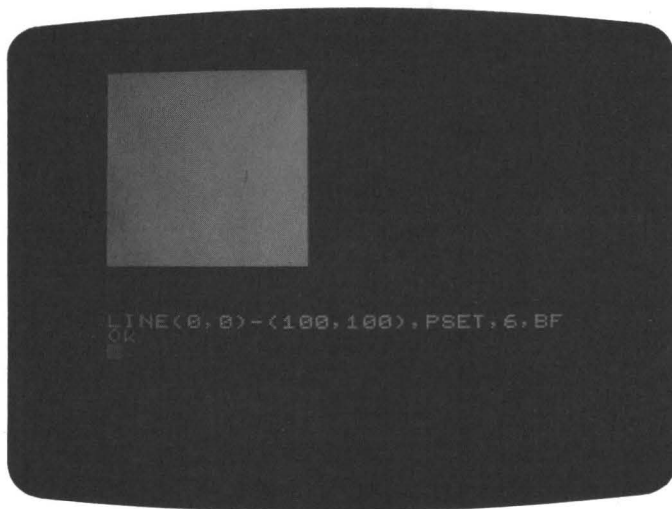
BOX というのは，LINE 命令とほぼ同じ。ひとつだけ違うのは，パレットナンバーの後にBと指定することだけです。

すると，指定した LINE を対角線とする長方形を描いてくれるんですよ。

（P.190のカラー写真参照）

わかりますか？（0, 0）から（100, 100）までひいた線が対かく線になってるでしょ。おもしろい命令だなあ。

BOX FULLはBOXの中をぬりつぶす命令で、BOXと違う指定の仕方はBがBFに

次に BOX FULL というのは，BOX の中をぬりつぶしてく

れる命令だそうです。指定の仕方は BOX とほとんど同じで，ただBを BF とすれば OK！

（P.190のカラー写真参照）

ワッきれい！これでいろいろな色の BOX がいっぱい描けるなあ。大きさもかえたりして……まあ遊んでみてください。

CIRCLEは円を描く命令です。中心と半径を指定するほか、y軸比率によっては楕円も…

CIRCLE は円を描く命令です。

CIRCLE(X, Y), R, C, y/x, A, B

指定の仕方はこの通りです。（☞78）

(X，Y)	……中心座標
R	……半径
C	……カラーナンバー
y／x	……半径の比率（yのxに対する半径）
A，B	……角度の指定（A度からB度まで）

半径の比率はx座標の半径1に対してyはいくつになるかということ。つまりきれいな形の円ばかりじゃなくて，だ円もかけるというわけです。

それに角度の指定があるっていうことは，扇形も自由自在というわけネ。これは，0度から60度まで指定してみました。

78

円を描く命令

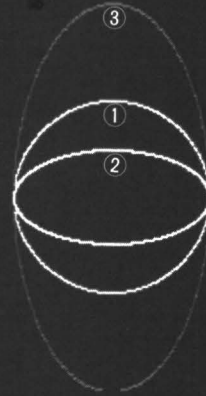
CIRCLE　サークル

CIRCLEは、画面にドットで円や弧を描くための命令です。

中心(X，Y)、半径r（ドット）の円が画面に描かれます。ゆりちゃんの説明のように、扁平率を設定すると、長軸がrのだ円を描くことができます。

また、初期角と終了角を指定すると、その内角の弧が描けます。円グラフなどを作製する時に役立ちますよ。皆さんも値を変えて色々な形を描いてみてください。

何も指定しなければ
普通の円になります

```
① CIRCLE (320,100),50,4
Ok
② CIRCLE (320,100),50,2,2
Ok
③ CIRCLE (320,100),50,5,.5
Ok
```

```
CIRCLE (320,100),70,7,1,0,60
Ok
```

（P.190のカラー写真参照）

POLYとは読んで字のごとく多角形、角数によって360/N(Nは角数)とします

POLY というのは，多角形を作る命令。

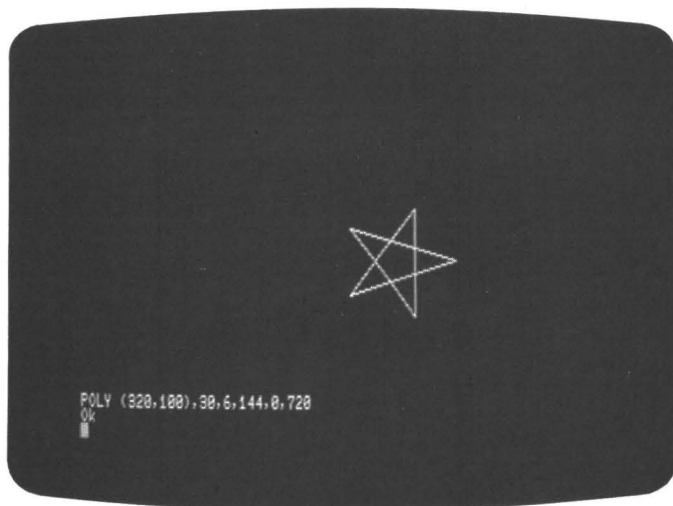
指定の仕方は CIRCLE に似てるみたい……。(☞79)

POLY(X, Y), R, C, 360/N, A, B

違うのは 360/N のところだけでしょ？このNは角の数で，5角形をつくりたければ5とすれば良いわけです。わかりますよね。ただ，原理からいくと，0度の点から何度おきに LINE を引くか（360/N がその角度）ということになるんだそうです。

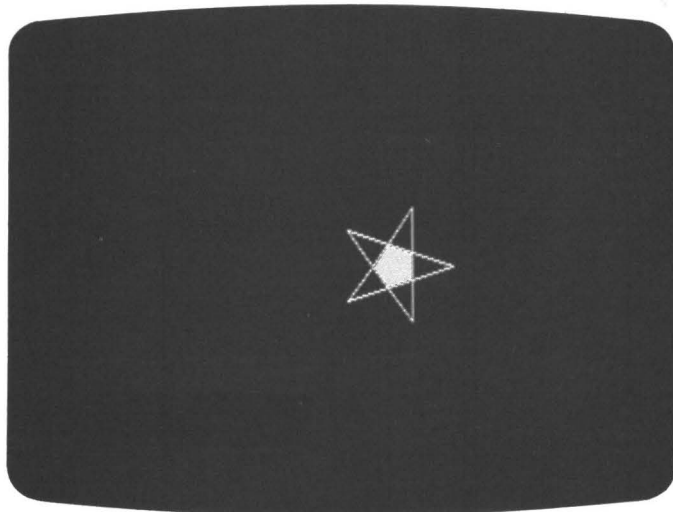
79

POLY　ポーリーと言い、多角形を描く命令です。

（x，y）を多角形の中心として、中心から各頂点までの距離をｒとした多角形を描きます。頂点間の角度をステップ角と呼び、120で三角形、90で四角形、☆は144となり、この角度が０の時は、中心から、初期角の方向へ長さｒの直線を引きます。

大成功！　（5角形を作るのに最後のA，B指定で0度〜720度まで回しています）

次に **PAINT** は想像通り，指定された領域をぬりつぶす命令。

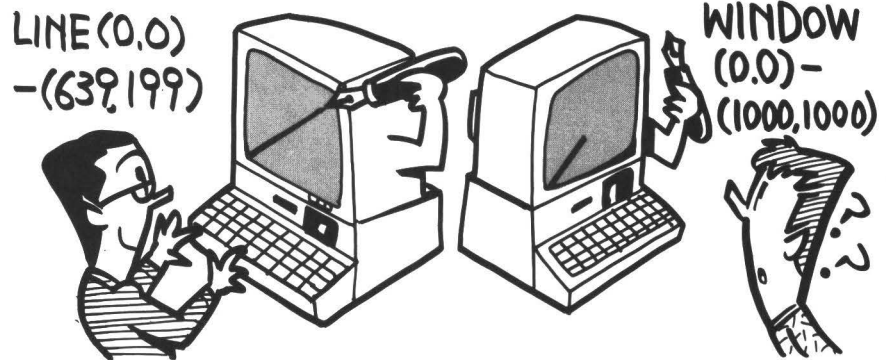
ただ，この命令はぬりつぶすだけのものですから，あらかじめ図形を描いておいて，その後使うものですね。

それじゃ，POLY の説明のところで描いた星形の図形の中央をぬりつぶしてみますね。(☞80)

（P.190のカラー写真参照）

こんな感じで，境界色を決め，その色にぶつかったら，ぬりつぶすのをやめてくれるわけですよね。ずいぶん器用なコンピュータだと思いません？

80

PAINT　　**ペイント命令**は、画面の特定のエリアを指定した色で塗りつぶしなさい、という命令です。

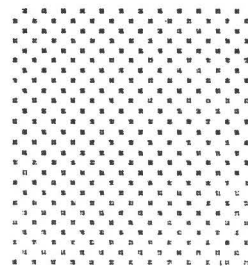
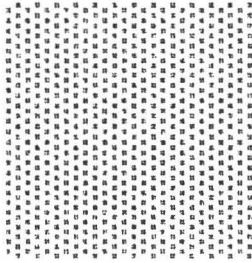
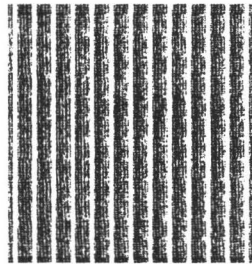
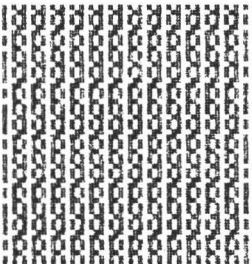
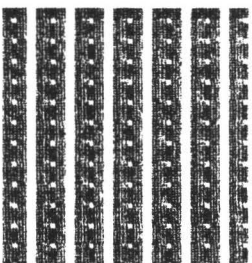
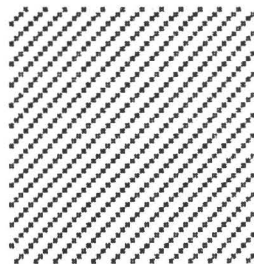
(x, y)の点を塗り始めの点として、境界となるパレットコードで指定された図形内を、指定のパレットコードで塗りつぶします。

さて，ここでおまけがついてくるんですけど，BOX FULLとか PAINT（後で出てきます）など，ある一定のスペースをぬりつぶす場合にちょっとおもしろい事があるんです。

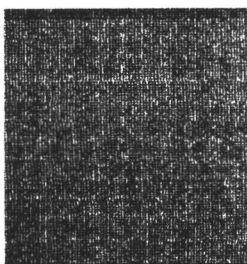
HEXCHR$は指定された範囲内を(" ")のパターンで塗りつぶす命令です

さて下のプログラムを見てください。計6種類のプログラムが，PRINT OUT されていますよね。

プログラムを見てわかるように，BF の後にHEXCHR＄（“ ”）と書いてあるのが分かるでしょう？

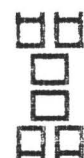
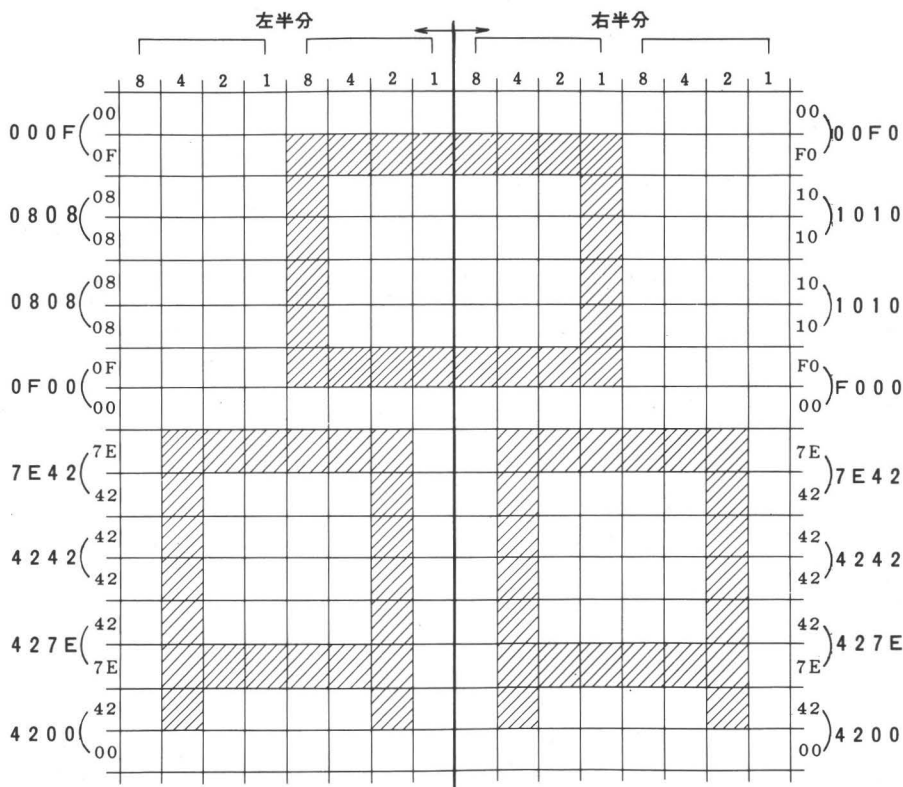
HEXCHR＄という命令は，指定された範囲内を，（“…”）のパターンを描く命令です。

```
line(50,50)-(100,100),pset,bf,hexchr$("1
111110000004444440000000")
Ok
■
```

```
line(50,50)-(100,100),pset,bf,hexchr$("8
888882222222888888222222")
Ok
■
```

```
line(50,50)-(100,100),pset,bf,hexchr$("2
244884422882244 88")
Ok
```

```
line(50,50)-(100,100),pset,bf,hexchr$("8
888222244444888822222")
Ok
```

```
line(50,50)-(100,100),pset,bf,hexchr$("1
234567812345678")
Ok
```

```
line(50,50)-(100,100),pset,bf,hexchr$("1
11111222222444444888888")
Ok
```

もちろん,通常のBOX FULLで行われることを,HEXCHR＄を使ってもできます。全部ぬりつぶしてしまうパターンは FFFF…です。

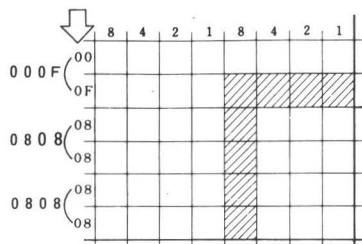
```
line(50,50)-(100,100),pset,bf,hexchr$("f
fffffffffffffffffffffff")
Ok
■
```

HEXCHR$を使って品という漢字を作ってみました。16進数を用います

今度は, このプログラムです。これは HEXCHR＄を使って品という漢字を作ったものです。

10番で品という字のパターンを作っています。

```
list
10 k$=HEXCHR$("000f080808080f007e4242424
27e420000f010101010f0007e424242427e4200"
)
20 PALET:POSITION 100,100:PATTERN-16,k$
30 POSITION 100,100:PATTERN 16,k$
Ok
run
Ok
■
```

この図を見てください。1つの文字を作るのに，このようにパターンを決めてやらねばなりません。この小さな四角1つが1 DOT です。HEXCHR＄を使う場合には，16進法が用いられるそうです。(☞81)

キャラクターを作る時は左半分から始めます。自分の名前に挑戦!!

図A

キャラクターを作る時は，まず左半分から始めます。8421はひと組みと考えて，DOT がぬりつぶされる部分は，この組み

81

HEXCHR$　**ヘキサキャラクターダラー**と呼びますが，我々は**ヘックスチャーダラー**と呼んでおります。

これは，16進数の文字データを文字列に変換します。

ゆりちゃんの説明のように，16進数の文字列を最初から2文字(2バイト)ずつ区切って，その2文字に対応するキャラクターに変換して文字列を作っていきます。この文字列が，この実数の値となるのです。

X1のオプションとして漢字ROMがありますが，このHEXCHR$を使うと自分で好きな漢字パターンを作ることも出来ますよ。この応用例は巻末の日本国憲法の中で出て来ます。参考にしてください。

ごとに1つの数字で表わしましょうネ。

左上をクローズアップして考えてみましょう。まず図A1段目を見てください。何もぬりつぶしがありませんよね。ですから00，次の2段目初めの4個は0，その後の4個はぬりつぶし，8+4+2+1=15=FとなるのでOF。次，3段目と4段目も左側の4個は0，その後は8の所だけぬりつぶすわけで，これは0808となります。

10番のリストを見てください。“000F，0808，0808……と16組ありますね。上の様にまず左上から左下まで8組計算し，次に右上から右下まで8組と順番に計算して行ったのです。

これがキャラクターを作る手順です。めんどうがらずに自分の名前を描いてみましょう。そのためには8，4，2，1の16進数計算を身につけてください。8＋4はC，8＋4＋1はD，8＋4＋2はE，8＋4＋2＋1はFということですよ。

POSITIONはテキスト画面のLOCATEに相当します。PATTERNも一緒に覚えましょう

さて，説明が長くなりましたけど，20番を解説しますネ。ここで，また新しい命令が2つ出てきます。

POSITIONとは，テキスト画面でいうLOCATEです。書き出す位置を指定してやるわけですね。POSITIONはグラフィック画面の時だけ使い，同様にLOCATEはテキスト画面にだけ使います。

次に **PATTERN** は POSITION と一緒に使います。20番を実行したものは“品”で，30番が“吅”です。どうしてこんな風になっちゃったんでしょう？

PATTERN −16

PATTERN　16

違いは，−がつくかつかないかにあるようですネ！　PATTERN −16では，POSITIONで指定された点から下へ16 DOT，

PATTERN 16では上へ16 DOT 書いていってくれるんです。

書くということは，さっきみたように DOT をぬりつぶすことで，その順番もキャラクターの作り方で書いた通りです。

普通は，上から下へと書いていきますから，大体の場合は PATTERN －○○となりますネ。

WINDOWはグラフィック画面の表示エリアを指定する命令で、対角線指示になります

さて，もう一つ重要なグラフィック・コマンドがありました。この **WINDOW** は，グラフィック画面の表示エリアを指定する命令です。考え方は少し難しいようですよ。(☞82)

```
window(0,0)-(639,199),(0,0)-(1000,1000)
Ok.
line(0,0)-(639,199),pset,7
Ok.
█
```

このプログラムを見てくださいね。(０，０)－(639，199) というのは最初のエリアの指定です。ここの考え方は，ボックスと同じで，(０，０)，(639，199) を結んだ線を対角線とした四角形がエリアとなります。

今，(０，０)－(639，199) を (０，０)－(1000，1000) と置き換えてみましょう…というのが **WINDOW 命令**なんです。普通に考えると，line (０，０)－(639，199) としたら，画面一杯に対角線状に線がひけるはずなんですよね。それがこのようになってしまったのは，(639，199) を (1000，1000)と置き直したためです。わかるかしら……？

WINDOW 命令は，画面の縮小や拡大，反転（座標系を逆にする）などにはとても便利です。

次のも見てみましょう。

82

WINDOW　そうです、この命令はグラフィックの窓を指定するのです。

窓、つまりグラフィック画面の表示エリアを設定する命令で、ユリちゃんの説明に補足いたしましょう。

WINDOW(Xs,Ys)-(Xe,Ye),(X1,Y1)-(X2,Y2)

Xs, Ys…始点座標

Xs…水平座標　40文字モードの時 0～319
80文字モードの時 0～639

Ys…垂直座標　0～199

Xe, Ye…終点座標

Xe…水平座標　40文字モードの時 0～319
80文字モードの時 0～639

Ye…垂直座標　0～199

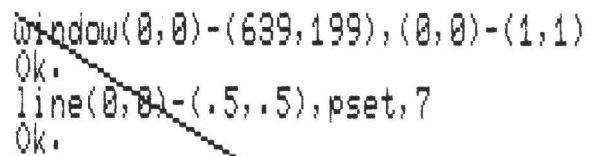
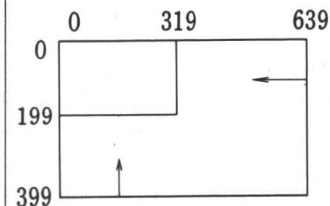
X_1, Y_1…始点の論理座標

X_2, Y_2…終点の論理座標

WINDOWのXs, YsからXe, Yeの位置は、実はX_1, Y_1からX_2, Y_2になるのですよ、と設定されるのです。

WINDOW(0,0)-(319,199),(0,0)-(319,199)

WINDOWの０，０から319, 199は、実は０，０から319, 199なのですよ。とすると、これは何の変化もありません。ところが、

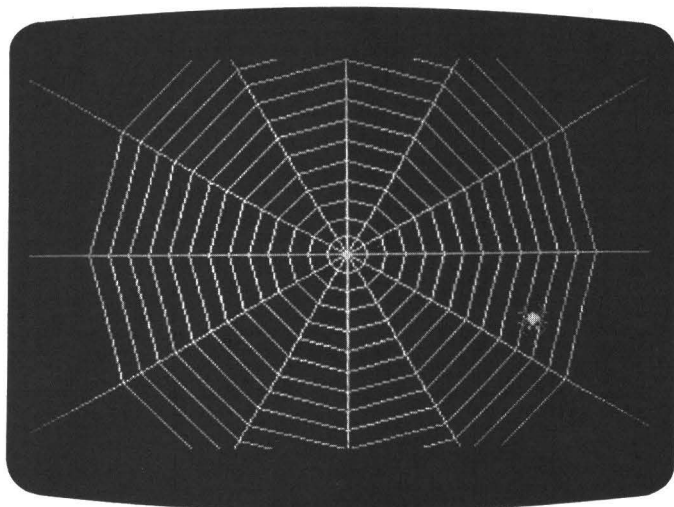
WINDOW(0,0)-(319,199),(0,0)-(639,399)

WINDOWの０，０から319, 199は、実は０，０から639, 399なのですよ。となるとどうなりますか。Xが639でYが399。ちょうど倍の数になってますね。ですから、画面の左上を基準に画面が¼になってしまったのです。

つまり、画面いっぱいにグラフィックを描いて、このWINDOWを実行すると、¼に縮まったグラフィックを画面上に描いてくれるわけですね。

このWINDOW命令は、巻末の応用例「世界の国々」の中で、イギリスの旗を縮めて、オーストラリアやニュージーランドを作るのに使われておりますよ。

line（0，0）—（.5，.5）なんて，点みたいな線が画面の隅に表示されるとばかり思ったら，こうなりました。(639,199)を（1，1）と設定し直したから，こうなったんですよ！

最初の方は，理論上の線よりも短かくなり，後の方は長くなったわけです。

くもの巣への応用

（それでは、グラフィック命令の応用としてくもの巣を描いてみましょう。このくもは動きますよ）

（P.190のカラー写真参照）

```
10 WIDTH40:SCREEN 0,0,0:PALET:PRW:CLS4:G
OSUB 180
20 FOR I=1 TO 12:POLY(160,100),200,7,0,I
*30:NEXT I
30 FOR I=1 TO13:POLY(160,100),I*10,7,30,
0,360:NEXT I
40 CGEN 1
50 LOCATE 20,13:PRINT"12";:LOCATE 20,14:
PRINT"34";
60 X=20:Y=13:XX=X:YY=Y
70 LOCATE X,Y:PRINT"12";:LOCATE X,Y+1:PR
INT"34";
80 SX=1:SY=1
90 X=X+SX:Y=Y+SY
100 IF X>34 THEN X=34:SX=-1
110 IF X<4 THEN X=4:SX=1
120 IF Y>23 THEN Y=23:SY=-1
130 IF Y<0 THEN Y=0:SY=1
140 LOCATE XX,YY:PRINT "  ";:LOCATE XX,Y
Y+1:PRINT "  ";
150 LOCATE X,Y:XX=X:YY=Y:PRINT"12";:LOCA
TE X,Y+1:PRINT"34";
160 FOR I=0 TO 1000:NEXT
170 GOTO 90
180 REM ｻﾌﾞﾙｰﾁﾝ
190 DEFCHR$(&H31)=HEXCHR$("08102010E9170
71F 08102010E917071F 0000000000000000")
200 DEFCHR$(&H32)=HEXCHR$("1008040897E8F
0F8 1008040897E8F0F8 0000000000000000")
210 DEFCHR$(&H33)=HEXCHR$("27CB112646201
008 27CB112040201008 0000000000000000")
220 DEFCHR$(&H34)=HEXCHR$("F4D3886462040
810 F4D3880402040810 0000000000000000")
230 DEFCHR$(&H20)=STRING$(24,0)
240 RETURN
```

83
さて、今回はグラフィックコマンドを駆使してクモの巣を描いてみました。このプログラム中に**PRW**、**CGEN**など勉強していない命令が多く出て来ますが、これらの命令は巻末の一覧表で勉強して下さい。(PRW…P 246参照、CGEN…P247参照)皆さんもX1HuBASICを勉強すると、この様な素晴しい絵を描くことができるのですよ。

LLISTはプリンタにLISTを打ち出し、LPRINTはプリンタにPRINT OUTしてくれます

次に，プリンタに関する命令を覚えておきましょうね。

○ LLIST

プリンターに LIST を打ち出してくれます。画面には LIST は出ません。

○ LPRINT

プリンターに PRINT OUT してくれます。これも画面には，何も表示されません。

```
llist
10 PRINT "Yuri "
20 PRINT "Shinagawa"
```

H COPYは文字だけの画面に書かれたことを全てプリントアウトしてくれます

```
10 print "Yuri  "
20 print "Shinagawa"
run
Yuri
Shinagawa
Ok
hcopy
```

○**HCOPY**（ハード・コピー）

①**HCOPY** TEXT画面，つまり文字だけの画面に書かれた事をすべてプリントアウトしてくれます。

②**HCOPY0** グラフィック・コマンドの中のスクリーン命令でも触れましたが，グラフィック・モードでは，TEXT画面，B，R，Gの4画面があります。このうちのB，R，G 3枚の画面とも，重ねて一つの画面として，コピーがとれるのが，この命令。ディスプレイ上には，黒を除いたすべての色（7色）が出てきます。

③**HCOPY1** B (Blue) の画面のみの表示となります。ですから，Bを構成要素として持つ色しか現われてきません。

④**HCOPY2** R (Red) の考え方は hcopy 1 と同じです。

⑤**HCOPY3** G (Green) の考え方も hcopy 1 と同じです。

⑥**HCOPY4** さて，これは TEXT画面と B，R，G 3枚のグラフィック画面を合わせた，計4枚の画面を合わせて表示し，プリンターに Print out する命令です。

①～⑥まで順番にプリントアウトしてありますから，見てみてくださいネ！

```
list
10 PALET
20 FOR i=0 TO 7
30 LINE (i*40,0)-(40*i+20,40),PSET,i,bf
40 NEXT i
Ok
run
Ok
hcopy
```

①

②

③

④

⑤

⑥

```
list
10 PA
20 FO     =0    7
30 LI     (i    ,0     40    20    ),    T,    bf
40 NE     i
Ok
run
Ok
hcopy 4
```

84

グラフィック命令のまとめ

SCREEN（GRAPH）

この命令は、グラフィック画面を使用するために必ず使われる命令です。この命令の使い方を間違いますと、画面が見えなくなったり、ＬＩＳＴも取れなくなります。そんな時の唯一の救助策を知って置いて下さい。

CTRL ＋ **D**（コントロールD）

この命令を実行することによって、なんと14個もの命令すべてが瞬時に実行され、もとの画面に戻ってしまいます。(P253参照)

ＳＣＲＥＥＮ命令の出力ページ、入力ページの指定は、40文字モードの時のみに有効で、80文字モードの時は入出力が同一画面となります。

COLORとPALETの再確略

カラーコード０～７とパレットコード０～７はどちらも同じゃないですか、と初めはゆりちゃんも考えていたようですが、これは
全く別のものなんですよ。

ＣＯＬＲ命令では、画面の文字の色と背景の色を変える命令でしたね。

ＣＯＬＯＲはHuBASIC起動時や、ＰＡＬＥＴ命令を使用しない時は…

COLORコード＝PALETコード

となり、指定通りのＣＯＬＯＲで変化してくれます。つまり全部の色が使える状態なんですね。

しかし、ＰＡＬＥＴ命令でキャラクター、つまり文字や背景の色を変化させることはできません。ただし、ＬＩＮＥ、ＬＩＲＣＬＥ、ＰＯＬＹなどのグラフィック命令を使用し、COLORコードを変化させないで、色に変化を与えることはできるのです。要するにグラフィック画面でのみ、このＰＡＬＥＴ命令が使えるのです。

LINE(0、0)－(319、199)、PSET、2
PALETコード

とダイレクトモードで実行してみてください。画面の左上から右下まで赤い線が描かれましたね。それではＰＡＬＥＴで色を変化させてみましょう。

PALET2.1

ＰＡＬＥＴコードの２(赤)を１(青)に変化させなさいと命令すると、ほら瞬時に青くなってしまったでしょう。

次に先ほどの……

CTRL ＋ D

と打ってみてください。赤に逆もどり初期状態にもどってしまったのです。

グラフィック処理に挑戦
PHOTOGRAPH集

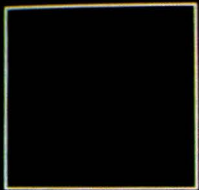
0ページから1ページに移すと、実はこんなことが書き込まれています

HuBASICのPALET機能、PALETナンバーに合わせて、それぞれブラック、ブルー、レッド、レッド、マゼンタ、グリーン、シアン、イエロー、ホワイトの8色表示ができます。

R,G,B三原色をディスプレイで表示すると、このようになります。

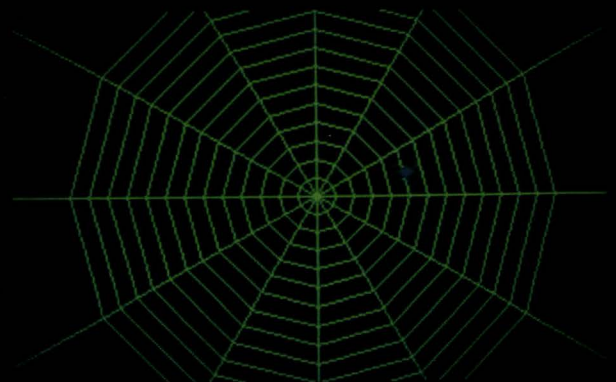
PSETで赤い点をXY座標上に表示。

LINE命令で破線を描きます。

←ランダムに点を表示させます。きれい!!

BOXからBOXFULLに指定を変えると、こんな風に美しくぬりつぶしてくれます。

CIRCLE命令では、こんな風にだ円を描くこともできますよ。

CIRCLE命令で0から60°まで描いてみました。

POLYで星を描き、PAINT命令で、中をぬりつぶします。

応用プログラムのくもの巣です。きれいでしょ!!

NOTE9
サンプル・プログラム
学習システム、図形処理、実用メモ、ミュージックの応用プログラム集

サンプルプログラムPHOTOGRAPH集

『日本国憲法』

美しい日本国旗とセイテイ、ハッコウの年月日が表示されます。

日本国憲法の前文。英語で書くとこうなります。

練習問題はこの11章の中から各1問。一回のチャレンジで、5問出題されます。

第1問は第1章「天皇」からですね。あなたは、できました。

Bee先生の場合は、5問中3問正解でした。

『世界の国々』

まず、この様な表示が出ますから、調べたい地域の番号をプッシュ。

すると、この様に各国のリストが表示。この中から更に調べたい国の番号をプッシュ。

アジア州

それでは、各国のディスプレイ表示をランダムに観て楽しみましょう。

日本

イエメン

イスラエル

イラク

インド

インドネシア

大韓民国

カンボジア

シリア

シンガポール

タイ

中華人民共和国

トルコ

パキスタン

バーレン

マレーシア

モルジブ

ヨルダン

タイヨウ州

オーストラリア

ツバル

トンガ

ニュージーランド

アフリカ州

アルジェリア

エチオピア

カメルーン

ギニア

ケニア

シエラレオネ

コモロ連邦

サントメプリンシペ

スーダン

セイシェル

セネガル

ソマリア

タンザニア

中央アフリカ

チエンジア

トーゴ

ベナン人民共和国

モーリタニア

モロッコ

リビア

リベリア

ヨーロッパ州 ソ連

アイスランド

イギリス

イエメン

アイルランド

イタリア

オーストリア

オランダ

ギリシア

スイス

スウェーデン

チェコスロバキア

デンマーク

ドイツ連邦共和国

ノルウエー

フランス

ベルギー

フィンランド

ユーゴスラビア

ソビエト連邦

北アメリカ州
南アメリカ州

アメリカ

カナダ

ジャマイカ

セントルシア

パナマ

バハマ

ホンジュラス

カイアナ

スリナム

チリ

ブラジル

全く何も知らない状態から，まだまだ不十分ではあるにせよ，一通りのコマンドやプログラムの考え方がわかるところまで来ましたヨ。皆さん，ご苦労さまでした。長い道のりでしたねぇ……ナンテネ！

さて，ここまで来たからには，私としては是非，直接のきっかけになった法律にコンピューターを結びつけたいのです。

最近，**日本国憲法**がベストセラーになったのは知ってますか？

THE VISUAL CONSTITUTION OF JAPAN

「あなたは読んだことがありますか」

と書いた赤帯が表紙についている本で，大きな活字と写真入りで読みやすくなっているんですよ。

1ページ目を開くと，地球を宇宙から見た写真がバーンと出ていて，日本列島の写真なんかも入っています。これは本当に写真かな。こうしてみると，日本って緑色の島なんですね。何故だかわからないけれど，ちょっと感動してしまったなあ……。ワッ！　今度は眼のアップ。大きく眼を開けて，自分の国の憲法を読んでごらんなさいというつもりなのかしら？

写真がきれいだし，アイディアもいいし，なるほど，ベストセラーになる要素は十分ですねぇ。六法の中の，ぎっしりつまった文字のイメージとは，まるで違いますもの。

さて，この日本国憲法を，プログラムにしようなんて，だいそれた考えかも知れませんね。でもともかくやってみましょう。

まず次頁の表をご覧下さい。

資料
THE VISUAL CONSTITUTION OF JAPAN 小学館発行

表1

日本国憲法練習問題

問　天皇は日本国の□□である。(第1条)

①総理大臣②元帥③象徴④恋人

問　国の□□はこれを認めない。(第9条―②)

①特権②実施権③国交権④交戦権

問　日本国民たる要件は□□でこれを定める。(第10条)

①憲法②民法③国会④法律

問　衆議院議員の任期は□□とする。(第45条)

①4年②1年③6年④3年

問　行政権は□□に属する。(第65条)

①国会②天皇③総理大臣④内閣

問　□□は最高裁判所の定める規則に従わなければならない。(第77条―②)

①裁判官②国民③皇室④検察官

問　国の財政を処理する権限は，□□の議決に基いて，これを行使しなければならない。(第83条)

①内閣②国会③裁判所④弁護士協会

問　地方公共団体には，法律の定めるところにより，その議事機関として□□を設置する。(第93条)

①地方裁判所②議会③地方自治体④振興会

問　憲法改正は，各議院の総議員の□□の賛成で，国会が，これを発議する。(第96条)

①5分の2②4分の1③3分の2④3分の1

問　憲法は，国の□□である。(第98条)

①財産②命③重要な法④最高法規

問　この憲法は，公布の日から起算して，□□を経過した日から，これを施行する。(第100条)

①1年②1週間③6箇月④3箇月

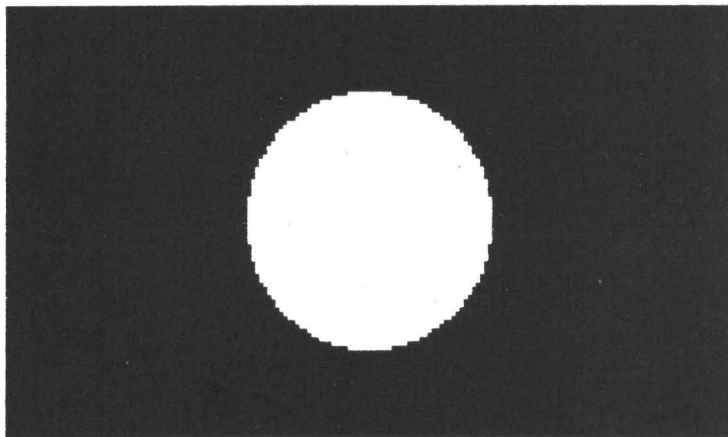
これは日本国憲法の第１章〜第11章の中から（第１章天皇，第２章戦争の放棄，第３章国民の権利及び義務，第４章国会，第５章内閣，第６章司法，第７章財政，第８章地方自治，第９章改正，第10章最高法規，第11章補則），ひとつずつ練習問題を出題したものです。

つまり，今作ろうとしているプログラムの内容は，これらの練習問題をプログラムにして，お友達の憲法に対する知識を試してみようというものなんですね。

ただ練習問題だけでは面白くないので，ついでに日本の国旗と英文で書かれた前文もプログラムしちゃいます。

最初にディスプレイとプログラムを示し，次にプログラムの説明をして行きますヨ。

THE CONSTITUTION OF JAPAN

日本国憲法

ｾｲﾃｲ:S21年11月3日 ﾊｯｺｳ:S22年5月3日

We,the Japanese people, acting through our duly elected representatives in the National Diet, determined that we shall secure for ourselves and our posterity the fruits of peaceful cooperation with all nations and the blessings of liberty throughout this land, and resolved that never again shall we be visited with the horrors of war through the action of government,do proclaim that sovereign power resides with the people and do firmly establish this Constitution. Government is a sacred trust of the people,the authority for which is derived from the people,

the powers of which are exercised by the representatives of the people, and the benefits of which are enjoyed by the people. This is a universal principle of mankind upon which this Constitution is founded. We reject and revoke all constitutions, laws, ordinances, and rescripts in conflict herewith.
We, the japanese people, desire peace for all time and are deeply conscious of the high ideals controlling human relationship, and we have determined to preserve our security and existence, trusting in the justice and faith of the peace-loving peoples of the world. We desire to occupy an honored place in an international society striving for the preservation of peace, and the banishment of tyranny and slavery, oppression and intolerance for all time from the earth.
We recognize that all peoples of the world have the right to live in peace, free from fear and want.
We believe that no nation is responsible to itself alone, but that laws of political morality are universal; and that obedience to such laws is incumbent upon all nations who would sustain their own sovereignty and justify their sovereign relationship with other nations.
We, the Japanese people, pledge our national honor to accomplish these high ideals and purposes with all our resources. RETURN ｷｰｦ ｵｼﾃｸﾀﾞｻｲ

第一問

ｸﾆ ﾉ ｻﾞｲｾｲ ｦ ｼｮﾘｽﾙ ｹﾝｹﾞﾝ ﾊ ▒▒▒▒ ﾉ ｷﾞｹﾂ
ﾆ ﾓﾄｽﾞｲﾃ､ｺﾚｦ ｺｳｼ ｼﾅｹﾚﾊﾞﾅﾗﾅｲ｡

1:ﾅｲｶｸ　　2:ｺｯｶｲ　　3:ｻｲﾊﾞﾝｼｮ
4:ﾍﾞﾝｺﾞｼｷｮｳｶｲ

ｾｲｶｲ ﾊ ﾅﾝﾊﾞﾝ ﾃﾞｽｶ ?

第四問

ｹﾝﾎﾟｳ ｶｲｾｲ ﾊ､ ｶｸ ｷﾞｲﾝ ﾉ ｿｳｷﾞｲﾝ ﾉ ▒▒▒▒ ﾉ
ｻﾝｾｲ ﾃﾞ､ｺｯｶｲ ｶﾞ ｺﾚｦ ﾊﾂｷﾞ ｽﾙ｡

1:5分 ﾉ 1　　2:4分 ﾉ 1　　3:3分 ﾉ 2
4:3分 ﾉ 1

ｾｲｶｲ ﾊ ﾅﾝﾊﾞﾝ ﾃﾞｽｶ ?

```
1 REM ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
2 REM ▒                          ▒
3 REM ▒  ﾆﾎﾝｺｸ ｹﾝﾎﾟｳ ﾚﾝｼｭｳ ﾓﾝﾀﾞｲ   ▒
4 REM ▒                          ▒
5 REM ▒ Copyright (C) 1982,10,01 ▒
6 REM ▒                          ▒
7 REM ▒ Hudsonsoft Yuri Shinagawa ▒
8 REM ▒                          ▒
9 REM ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
10 DEFINT a-z:DIM ka$(52),m$(11),t(11)
20 WINDOW(0,0)-(319,199):SCREEN1,1,0:CLS0:PALET:WIDTH40:CLS:GOSUB 260
30 LINE(85,38)-(190,58),PSET,3,b:LOCATE 5,2:COLOR6:PRINT"THE CONSTITUTION OF JAP
AN":LINE(45,70)-(225,170),PSET,7,BF:CIRCLE (135,120),30,2:PAINT(135,120),2,2
40 FOR i=1 TO 5:POSITION i*20+70,40:COLOR 4,0:PATTERN-16,ka$(i):NEXT
50 LOCATE 2,23:PRINT "ｾｲﾃｲ:S21年11月3日    ﾊｯｺｳ:S22年5月3日":GOSUB 910:GOSUB 1000
```

```
60 PALET:CLS0:CLS:WIDTH80:COLOR4:po=622:p1=59:p2=40:FOR i=1 TO 11:COLOR 4
70 POSITION po,5:PATTERN-16,ka$(37):POSITION po,22:PATTERN-16,ka$(p2):POSITION p
o,39:PATTERN-16,ka$(38):po=po-p1:p2=p2+1
80 FOR j=1 TO 9:READ yk:COLOR 6:POSITION po+29,21+j*18 :PATTERN-16,ka$(yk):NEXTj
,i:FOR i=1 TO 5000:NEXT i:COLOR7
90 LINE(0,140)-(400,198),PSET,7,b:LOCATE 2,18:PRINT "ﾆﾎﾝｺｸ ｹﾝﾎﾟｳ ｦ ﾀﾀﾞｼｸ ﾘｶｲ ｽﾙﾀ
ﾒ ﾆ ｶｸ ｼｮｳ ﾉ ﾓﾝﾀﾞｲ ":PRINT "  ﾆ ｺﾀｴﾃ ｲﾀﾀﾞｷ ﾏｽ｡ ﾀﾞｲ 1 ｼｮｳ ｶﾗ ﾀﾞｲ 11 ｼｮｳ ﾏﾃﾞﾉ ":PR
INT"  ﾅｶｶﾗ ﾑｻｸｲ ﾆ 5 ﾓﾝ ｼﾂﾀﾞｲ ｲﾀｼﾏｽ､ ｾｲｶｲ ｦ ｴﾗﾝﾃﾞ ｸﾀﾞ""
100 PRINT"  ｻｲ｡ ｱﾅﾀ ﾉ ｹﾝﾎﾟｳﾆ ﾀｲｽﾙ ﾘｶｲﾄﾞ ｦ ﾃｽﾄ ｻｾﾃ ｲﾀﾀﾞｷﾏｽ｡"
110 CFLASH 1:LOCATE 15,23:PRINT " RETURN ｷｰｦ ｵｼﾃｸﾀﾞｻｲ ":WHILE INKEY$="":WEND:CFL
ASH 0
120 FOR i=1 TO 11:READ m$(i):NEXT i
130 FOR i=1 TO 11:t(i)=0:NEXT i:p=0:FOR i=1 TO 5
140 x=INT(11*RND(1))+1:xx=t(x):IF xx<>0 THEN 140 ELSE t(x)=1
150 CLS0:CLS:PALET:WIDTH 40:COLOR 5
160 POSITION 5,5:PATTERN-16,ka$(37):POSITION 23,5:PATTERN-16,ka$(39+i):POSITION
41,5:PATTERN-16,ka$(51):COLOR 4
170 LOCATE 0,5:PRINT LEFT$(m$(x),68):LOCATE 3,10:PRINT MID$(m$(x),69,55)
180 COLOR 7:LOCATE 3,15:PRINT "ｾｲｶｲ ﾊ ﾅﾝﾊﾞﾝ ﾃﾞｽｶ ";:z$=INKEY$(1):IFz$=""THEN180
ELSE z=VAL(z$)
190 IF z=VAL(RIGHT$(m$(x),1))THEN p=p+1:LOCATE 23,15:CFLASH 1:PRINT "ｾｲｶｲ !!" EL
SE LOCATE 23,15:PRINT "ﾏﾁｶﾞｲ.."
200 LOCATE 6,23:CFLASH 1:PRINT "RETURN ｷｰｦ ｵｼﾃｸﾀﾞｻｲ｡":WHILE INKEY$="":WEND:CFLAS
H 0:NEXT i
210 CLS0:CLS:PALET:COLOR 4:POSITION 100,50:PATTERN-16,ka$(39+p):POSITION 118,50:
PATTERN-16,ka$(51):POSITION 136,50:PATTERN-16,ka$(31):POSITION 154,50:PATTERN-16
,ka$(52)
220 COLOR 6:LOCATE 15,11:PRINT "ﾃﾞｼﾀ｡"
230 LOCATE 3,15:PRINT "ﾓｳｲﾁﾄﾞ ﾁｮｳｾﾝ ｼﾏｽｶ (y or n) ";
240 a$=INKEY$(1):IF a$="" THEN 240 ELSE IF a$="y" THEN 130
250 CLS0:CLS:END

260 ka$(1)=HEXCHR$("001f10101010101f101010101010101f0000fc04040404040fc040404040404
fc00"):'ﾆ
270 ka$(2)=HEXCHR$("0000007f0001020204080810274000000808080ff80c0a0a090888884f281
8080"):'ﾎﾝ
280 ka$(3)=HEXCHR$("003f20202f20202027202020202f20203f00fe0202fa828282f282a292fa02
02fe"):'ｺｸ
290 ka$(4)=HEXCHR$("007f405f000f007f003f223f0024244380ff81fd80f880ff00fe22fe0082
49f1"):'ｹﾝ
300 ka$(5)=HEXCHR$("2010080340201007000809111222274020202 0fe202020ff80800004021e
e101"):'ﾎﾟｳ
310 ka$(6)=HEXCHR$("007f00000003f010101020204081060 00ff808080 80fe404040202010 08
0403"):'ﾃﾝ
320 ka$(7)=HEXCHR$("010f08080f08080f003f00001f00007f00f80808f80808f800fe8080fc80
80ff"):'ﾉｳ
330 ka$(8)=HEXCHR$("49292a007f49497f49497f08087f0808101412101f70121212121414 0809
35c3"):'ｾﾝ
340 ka$(9)=HEXCHR$("02030c00001f00007f00001f0000000300f0102040fc8484ff8484fc8080
8080"):'ｿｳ
350 ka$(10)=HEXCHR$("0000010608101021212122120c0000000000e098848282020202040418e
```

```
00000"):'ﾉ
360 ka$(11)=HEXCHR$("0808087f1010101e121212122222224c2020207f42a2222214140808142
44281"):'ﾎｳ
370 ka$(12)=HEXCHR$("007f04093e087f08080f007f020c700080ff08f40288ff8888f880ffa09
88780"):'ｷ
380 ka$(13)=HEXCHR$("001f1010101f10101f10101010101f7000fc040404fc8080ff404020201
10d03"):'ﾐﾝ
390 ka$(14)=HEXCHR$("101010107d1011383450511210101010404040fea02020ff4448ff88fe88f
e88ff"):'ｹﾝ
400 ka$(15)=HEXCHR$("033c0404047f040c0e152444040404040202222222a2222222222222020
2020e"):'ﾘ
410 ka$(16)=HEXCHR$("001f0202020205050504080810102043 00f0101020203c0404888850202
0d807"):'ｵﾖ
420 ka$(17)=HEXCHR$("00000244380810102020202020100f0002012422303028262020204040 8
00000"):'ﾋﾞ
430 ka$(18)=HEXCHR$("04023f001f007f003f047f04077c041c1020fe80fc80ff20a824fe10940
935c3"):'ｷﾞ
440 ka$(19)=HEXCHR$("007e042910087e0b1a18294808080 81b407f844428106c832020fe22424
2820c"):'ﾑ
450 ka$(20)=HEXCHR$("0001020408102740000 03f010202043f804020100804f2010000fe00080
47a82"):'ｶｲ
460 ka$(21)=HEXCHR$("00003f20202121212224282020202020 8080fe8282424242221 20a02020
2020e"):'ﾅｲ
470 ka$(22)=HEXCHR$("003f213f213f212122252023262a2320007e427e427e02e2224282e2322
ae20e"):'ｶｸ
480 ka$(23)=HEXCHR$("007f00003f00001f101010101f00000000fe0202f20202c242424242c20
2021e"):'ｼ

490 ka$(24)=HEXCHR$("003f21213f21213f2121213f00142241040404047f040c141414242444 8
4041c"):'ｻﾞｲ
500 ka$(25)=HEXCHR$("007f040424242427242424242 4277c001090103f222242222414140 8081
422c1"):'ｾｲ
510 ka$(26)=HEXCHR$("10101010107c1013101014183040000010109090909ef29292929690918
1817e"):'ﾁ
520 ka$(27)=HEXCHR$("0000007f020202030204040408081060808080ff000000fc04040404040
40830"):'ﾎｳ
530 ka$(28)=HEXCHR$("0001021f1010101f1010101f1010101f800000fc040404fc040404fc040
404fc"):'ｼﾞ
540 ka$(29)=HEXCHR$("201008004020130000090911112121412020404482 8df10000fe0202020
202fe"):'ﾁﾞ
550 ka$(30)=HEXCHR$("007c0404047c41404044444444438000140407f428282424444242828102
84483"):'ｶｲ
560 ka$(31)=HEXCHR$("003f000000000808080808080808087f00fe40404040407e40404040404
040ff"):'ｾｲ
570 ka$(32)=HEXCHR$("001f101f101f007f111f111f1117790100fc04fc04fc00ff007e2222140
81463"):'ｻｲ
580 ka$(33)=HEXCHR$("00007f000f08080f003f2027242427208080ff00f80808f800fe02f2121
2f206"):'ｺｳ
590 ka$(34)=HEXCHR$("0808083e0808087f081414222220400700fe8282fe8282fe8282fe28484
9890f"):'ｷ
600 ka$(35)=HEXCHR$("1010117c040408081a3452101010101610ff1010fe9292fe9292fe929
```

```
29296"):'ﾎ
610 ka$(36)=HEXCHR$("003f20203f2020203f2020203f0011600282929292929292929292820
202ce"):'ｿｸ
620 ka$(37)=HEXCHR$("101f2442001f00001f10203f020c7000203f488400fc8484fc8080fe828
28c80"):'ﾀﾞｲ
630 ka$(38)=HEXCHR$("00003f04027f001f101f101f007f00008080fe1020ff00fc04fc04fc80f
f8080"):'ｼﾞｮｳ
640 ka$(39)=HEXCHR$("00030c10102020202020201010 0c030000e0180404020202020202040 41
8e000"):'0
650 ka$(40)=HEXCHR$("0000000000007f000000000000000000000000000ff0000000000
00000"):'1
660 ka$(41)=HEXCHR$("00001f000000000000000007f0000000fc00000000000000000000f
f0000"):'2
670 ka$(42)=HEXCHR$("00003f0000000001f0000000007f000000fe000000000fc00000000
0ff00"):'3
680 ka$(43)=HEXCHR$("003f22222222224242428302020 3f0000fe42424242424242423e02020
2fe00"):'4
690 ka$(44)=HEXCHR$("003f01010101011f02020202040404 7f00fe000000000fc04040404040
404ff"):'5
700 ka$(45)=HEXCHR$("00000007f0000000040408081020408080808 0ff000000000808040 40
20202"):'6
710 ka$(46)=HEXCHR$("02020202020237e020202020202020100000000007f80000000000010
101fe"):'7

720 ka$(47)=HEXCHR$("000202020204040404080808101020400020202020101010080808040
40201"):'8
730 ka$(48)=HEXCHR$("02020202 3f02020202040404080810600000000f01010101010101 1111
1111f"):'9
740 ka$(49)=HEXCHR$("000000000000 7f000000000000000080808080808080ff80808080808
08080"):'10
750 ka$(50)=HEXCHR$("000000000000 7f0000000000000003f008080808080 80ff80808080808
000fe"):'11
760 ka$(51)=HEXCHR$("003f213f213f20202724242427202020007e427e427e0202f2121212f20
20206"):'ﾓﾝ
770 ka$(52)=HEXCHR$("101f21423f24243f24243f2020204043003f091191a7809494bfa4c484b
f8484"):'ｶｲ
780 RETURN

790 DATA 6,7,0,0,0,0,0,0,8,9,10,11,12,0,0,0,0,3,13,10,14,15,16,17,18,19,3,20,0
,0,0,0,0,0,0,21,22,0,0,0,0,0,0,23,5,0,0,0,0,0,0,24,25,0,0,0,0,0,0,0,26,27,28
,29,0,0,0,0,0,30,31,0,0,0,0,0,0,32,33,5,34,0,0,0,0,0,35,36,0,0,0,0,0,0,0

800 DATA"      ﾃﾝﾉｳ ﾊ ﾆﾎﾝ ｺｸ ﾉ ▒▒▒▒▒ ﾃﾞｱﾙ｡                                  1:ｿ
ｳﾘﾀﾞｲｼﾞﾝ   2:ｹﾞﾝｽｲ      3:ｼｮｳﾁｮｳ     4:ｺｲﾋﾞﾄ         3"
810 DATA"      ｸﾆ ﾉ ▒▒▒▒▒ ﾊ ｺﾚｦ ﾐﾄﾒﾅｲ｡                                       1:ﾄ
ｯｹﾝ          2:ｼﾞｯﾁｹﾝ     3:ｺｯｺｳｹﾝ     4:ｺｳｾﾝｹﾝ        4"
820 DATA"ﾆﾎﾝ ｺｸﾐﾝ ﾀﾙ ﾖｳｹﾝ ﾊ ▒▒▒▒▒ ﾃﾞｺﾚ ｦ ｻﾀﾞﾒﾙ｡                             1:ｹ
ﾝﾎﾟｳ         2:ﾐﾝﾎﾟｳ      3:ｺｯｶｲ       4:ﾎｳﾘﾂ          4"
830 DATA"    ｼｭｳｷﾞｲﾝ ﾉ ﾆﾝｷ ﾊ ▒▒▒▒▒ ﾄ ｽﾙ｡                                    1:4
年           2:1年        3:6年        4:3年           1"
840 DATA"    ｷﾞｮｳｾｲｹﾝ ﾊ ▒▒▒▒▒ ﾆ ｿﾞｸｽﾙ｡                                      1:ｺ
```

```
ｯｶｲ         2:ﾃﾝﾉｳ       3:ｿｳﾘﾀﾞｲｼﾞﾝ    4:ﾅｲｶｸ        4"
850 DATA"▒▒▒ ﾊ ｻｲｺｳｻｲﾊﾞﾝｼｮ ﾉ ｻﾀﾞﾒﾙ ｷｿｸ ﾆ ｼﾀｶﾞｳ  ﾅｹﾚﾊﾞﾅﾗﾅｲ｡                     1:ｻ
ｲﾊﾞﾝｶﾝ      2:ｺｸﾐﾝ       3:ｺｳｼﾂ         4:ｹﾝｻﾂｶﾝ       4"
860 DATA"ｸﾆ ﾉ ｻﾞｲｾｲ ｦ ｼｮﾘｽﾙ ｹﾝｹﾞﾝ ﾊ ▒▒▒ ﾉ ｷﾞｹﾂ  ﾆ ﾓﾄｽﾞｲﾃ､ｺﾚｦ ｺｳｼ ｼﾅｹﾚﾊﾞﾅﾗﾅｲ｡1:ﾅ
ｲｶｸ         2:ｺｯｶｲ       3:ｻｲﾊﾞﾝｼｮ      4:ﾍﾞﾝｺﾞｼｷｮｳｶｲ  2"
870 DATA"ﾁﾎｳ ｺｳｷｮｳﾀﾞﾝﾀｲ ﾆﾊ､ﾎｳﾘﾂ ﾉ ｻﾀﾞﾒﾙﾄｺﾛﾆﾖﾘ､ ｿﾉｷﾞｼﾞｷｶﾝ ﾄｼﾃ ▒▒▒ ｦ ｾｯﾁ ｽﾙ｡  1:ﾁ
ﾎｳｻｲﾊﾞﾝｼｮ   2:ｷﾞｶｲ       3:ﾁﾎｳｷﾞﾁﾀｲ     4:ｼﾝｺｳｶｲ       2"
880 DATA"   ｹﾝﾎﾟｳ ﾊ､ ｸﾆ ﾉ ▒▒▒ ﾃﾞ ｱﾙ｡                                            1:ｻ
ﾞｲｻﾝ        2:ｲﾉﾁ      3:ｼﾞｭｳﾖｳ ﾅ ﾎｳ   4:ｻｲｺｳﾎｳｷ      4"
890 DATA"ｹﾝﾎﾟｳ ｶｲｾｲ ﾊ､ ｶｸ ｷﾞｲﾝ ﾉ ｿｳｷﾞｲﾝ ﾉ ▒▒▒ ﾉ ｻﾝｾｲ ﾃﾞ､ｺｯｶｲ ｶﾞ ｺﾚｦ ﾊﾂｷﾞ ｽﾙ｡1:5
分 ﾉ 1       2:4分 ﾉ 1     3:3分 ﾉ 2      4:3分 ﾉ 1      3"
900 DATA"ｺﾉ ｹﾝﾎﾟｳ ﾊ､ ｺｳﾌ ﾉ ﾋ ｶﾗ､ ｷｻﾝ ｼﾃ､ ▒▒▒ ｦ  ｹｲｶ ｼﾀ ﾋ ｶﾗ､ ｺﾚ ｦ ｼｯｺｳ ｽﾙ｡  1:1
年          2:1ｼｭｳｶﾝ     3:6ｶ月         4:3ｶ月         3"
910 TEMPO100:PLAY"o4a4r1a4r1b7a4r1a4r1b7:o4e5e5b3o5e3c3o4b3e5e5b3o5e3c3o4b3"
920 PLAY"o4a5b5o5c5o4b5a5b3a3f6r3:o4r3e3r3e3r3e3r3e3r7r3d3o3b3o4d3"
930 PLAY"o4e5c5e5f5e5e3c3o3b6r3:o3r3a3r3a3r3a3r3a3r3a2r0a2r0a2r0e6r3"
940 PLAY"o4a5b5o5c5o4b5a5b3a3f6r3:o4r3e3r3e3r3e3r3e3r7r3d3o3b3o4d3"
950 PLAY"o4e5c5e5f5e5e3c3o3b6r3:o3r3a3r3a3r3a3r3a3r3a2r0a2r0a2r0e6r3"
960 PLAY"o4a4r1a4r1b7a4r1a4r1b7:o4e5e5b3o5e3c3o4b3e5e5b3o5e3c3o4b3"
970 PLAY"o4d5e5f6r0a2b3a3f8:o3a9a9"
980 PLAY"o4e7r0e5r0d5e9e5:o4c7r0c5r0o3a5b9b5:o3a7r0a5r0f5e9e5"
990 RETURN
1000 CLS0:CLS:WIDTH 80
1010 PRINT"  We,the Japanese people, acting through our duly elected representat
ives in the";
1020 PRINT"National Diet, determined that  we shall secure  for ourselves and ou
r posterity";
1030 PRINT"the fruits of peaceful cooperation with all nations and the blessings
 of liberty";
1040 PRINT"throughout this land, and resolved that never again shall we be visit
ed with the";
1050 PRINT"horrors of war through the action of government,do proclaim that sove
reign power";
1060 PRINT"resides with the people and do firmly establish this Constitution. Go
vernment is";
1070 PRINT"a sacred trust of the people,the authority for which is derived from
the people,";
1080 PRINT"the powers of which are exercised by the representatives of the peopl
e,  and the";
1090 PRINT"benefits of which are enjoyed by the people.    This is a universal p
rinciple of";
1100 PRINT"mankind upon which this  Constitution is founded.     We  reject and
revoke  all";
1110 PRINT"constitutions, laws, ordinances, and rescripts in conflict herewith."
1120 PRINT"  We, the japanese people, desire peace for all time and are deeply c
onscious of";
1130 PRINT"the high ideals  controlling  human  relationship,  and  we have  det
ermined  to";
1140 PRINT"preserve our security and existence, trusting  in  the  justice and f
aith of the";
```

```
1150 PRINT"peace-loving peoples of the world.  We desire  to occupy an honored place in an";
1160 PRINT"international society striving for the preservation of peace, and the banishment";
1170 PRINT"of tyranny and slavery, oppression and intolerance for all time from the earth.";
1180 PRINT"  We recognize that all peoples of  the world have the right  to live in peace,";
1190 PRINT"free from fear and want."
1200 PRINT"  We believe  that no nation  is responsible to itself alone,   but that laws of";
1210 PRINT"political morality are universal;  and  that obedience to such laws is incumbent";
1220 PRINT"upon  all nations  who would sustain  their own sovereignty  and  justify their";
1230 PRINT"sovereign relationship with other nations."
1240 PRINT"  We, the Japanese people,  pledge  our national honor  to accomplish these high";
1250 PRINT"ideals and purposes with all our resources.";
1260 COLOR 7:CFLASH 1:LOCATE 57,24:PRINT" RETURN ｷｰｦ ｵｼﾃｸﾀﾞｻｲ ";:WHILE INKEY$=""
:WEND:CFLASH 0:RETURN
Ok
```

さて日本国憲法練習問題プログラムの解説をしましょう。

１番～９番は **REM 文**で，Program のタイトルと製作年月日，製作者の名前などを入れておきます。REM は**注釈**でしたよね。

10　プログラムの最初には，必ず初期化が必要です。

DEFINT a—z

すべての変数を整数形，つまりインテジャーで使用するということを宣言します。

DIM Ka$ (52), m$ (11), t (11)

プログラムの中で使用される配列変数を宣言します。

Ka$ (52)＝漢字パターン52個の配列

m$ (11)＝練習問題11題の配列

t (11)＝同じ問題を２度繰り返さないための配列

20　画面の初期化を行います。

WINDOW（0, 0）—（319, 199）

画面全体にグラフィックが出せる指定です。

SCREEN 1, 1, 0

ディスプレイに表われる画面が1で，プログラムの書かれる画面が1，グラフィックでは3面つかえるという指定です。

CLS 0

グラフィック画面を全て消します。前に描いてあったグラフィック画面を消すためです。

PALET

このプログラムでは，グラフィックを使えますよ。

WIDTH 40

40文字モードの画面を使います。

CLS

テキスト画面も全部消しましょう。

GOSUB 260

260番の サブルーチンに飛びます。

260 **260番から770番までは，漢字のグラフィック・パターンが入っています。**

Ka$ (1) = HEXCHR$ ("……")

漢字のパターンは HEXCHR$ のところで説明した通り，16×16 DOT のパターンデータで構成されています。

780 RETURN

サブルーチンをぬけます。

30 **最初のグラフィック画面をつくります。**

LINE (85, 38) — (190, 58), PSET, 3, b

(85, 38) から (190, 58) までの線を対角線にした BOX をマゼンダで描きます。

LOCATE 5, 2

(5, 2) の座標から

COLOR 6

黄色で

PRINT "THE CONSTITUTION OF JAPAN"

THE CONSTITUTION OF JAPAN と書きます。

LINE (45, 70) — (225, 170), PSET, 7, BF

（45，70）から（225，170）までの線を対角線とするBOX を，白でぬりつぶす。

CIRCLE（135，120），30，2

赤い半径30の輪を，（135，120）を中心点として描きます。

PAINT（135，120）2，2

先に描かれた赤い輪まで，同じく赤い色でぬりつぶします。これで日の丸のでき上がり！

40　日本国憲法と，漢字で前に描いたマゼンダの BOX の中に入れましょう。

FOR i＝1 TO 5

漢字の数は５個だからね。

POSITON i＊20＋70，40

X軸を１文字20 DOT ずつずらしましょう。

COLOR 4，0

文字の色は緑色にします。

PATTERN－16，Ka＄（i）～NEXT i

５個の漢字ができ上がり。

50　最後にメッセージを書きましょう。

LOCATE 2，23

書き出しは２，23です。

PRINT "セイテイ，ハッコウ"

制定　昭和21年11月３日，発効　昭和22年５月３日

漢字が簡単に書けたら，これも漢字にしたかったのにネ。

GOSUB 910

910行のサブルーチンへ

910　音楽『さくらさくら』の サブルーチン

TEMPO 100

TEMPO はやや遅めにしました。

PLAY

これが苦労の三重奏，日本調を出すためお琴風の演奏で……。

990　RETURN

50 番へまい戻り。

50　GOSUB 1000

またまたサブルーチンなのね。

1000　CLS 0 : CLS : WIDTH 80

画面を全て消して，80文字モードにいたしましょう。

1010　**日本国憲法前文。**ちょっと気どって英語で書いてみました。

“日本国民は，正当に選挙された国会における代表者を通じて……”

1250　“日本国民は，国家の名誉にかけ，全力をあげてこの崇高な理想と目的を達成することを誓ふ”。

1260　**読み終わったら合図を送りましょう。**

COLOR 7

文字は白で。

CFLASH 1

フラッシング・キャラクターを使って。

LOCATE 57, 24

57，24の位置に，

PRINT “RETURN キーヲオシテクダサイ”

とメッセージ！

WHILE INKEY$ = “ ” : WEND

キー入力があるまで，待ちましょう。

CFLASH 0

フラッシングキャラクターを止めましょう。

60　**目次をグラフィックでかっこ良く。**

PALET

またまたグラフィック・モードに戻しましょう。

CLS 0 : CLS

前の画面を全て消します。

WIDTH 80

画面はフルに使いましょうね。

COLOR 4

色は緑で

Po＝622

グラフィックの POSITION を変数 PO に入れました。

P 1＝59

各章は，59 DOT ずつあけましょう。

P 2＝40

使う漢字は40個です。

FOR i＝1 TO 11

第１章から第11章

COLOR 4

緑の色で

70 **“第○章”を書くルーチン**

POSITION PO, 5

画面の右から書きましょう。

PATTERN—16, Ka$ (37)

“第”

POSITION PO, 22

数字をかく位置

PATTERN—16, Ka$ (P 2)

“□”

POSITION PO, 39

章を書く位置

PATTER—16, Ka$ (38)

“章”

PO＝PO—P 1

POSITION をずらして下さい。

P 2＝P 2＋1

次の漢字配列の準備！

80 **各章の内容を入れましょう。**

FOR j＝1 TO 9

一行に９文字入れましょう。

READ yk

変数 yk に漢字データを読み込みます。

790 DATA

DATA は全部で 9 × 11 個（99個）。

6， 7， 0， 0， 0， 0， 0， 0， 0

たとえば“天皇”では漢字データの 6 番（310行）に天，7 番（320行）は皇，後は 7 個分のスペースとなります。なぜスペースを書いたかというと，一番長い文章は，第 3 章の“国民の権利及び義務”の 9 文字だから，全て 9 文字分のデータを入れたというわけ！　一度使った DATA は 2 度と使用できません。何度も使おうとするためには，変数 yk を配列変数にしておけば良いですネ。この辺は，皆さんお好きなように改造してみてください。

80 COLOR 6

内容は黄色で。

POSITION PO＋29， 21＋j＊18

横位置は PO＋29，縦位置は 21＋j ＊ 18

一字は16×16 DOT なので，2 DOT あけて 18 DOTずつ並べましょう。

PATTERN—16， Ka$（yk）

今の指定に，一文字入れて。

NEXT j， i

全ての漢字ができ上がり！　NEXT j は80番の頭のFOR i で，これは内容。NEXT i は40番の頭の FOR i で，これは全体。全体の FOR i，NEXT i の中に，内容の FOR j, NEXT j があるというわけです。

FOR i＝1 TO 5000：NEXT i

実はこれタイマー。5000数えるまで待ちましょう!!

COLOR 7

色は白に変えて。

90 PRINT 文でメッセージ！

LINE（0， 140）—（400， 198）， PSET， 7， B

画面の左下に白い BOX を描きましょう。

LOCATE 2, 18

書き出しは BOX の中。

PRINT "ニホンコクケンポウヲタダシクリカイスルタメニカクショウノモンダイ"

PRINT "ニコタエテイタダキマス。ダイ1ショウカラダイ11ショウマデノ"

PRINT "ナカカラムゾウサニ5モンシツダイイタシマス。セイカイヲエランデクダ"

100 PRINT "サイ。アナタノケンポウニタイスルリカイドヲテストサセテイタダキマス"

と4行分のメッセージ。

110 **KEY入力を待ちましょう。**

CFLASH 1

フラッシング・キャラクターを使いましょう。

LOCATE 15, 23

一番最後に，PRINT "RETURN キーヲオシテクダサイ" とメッセージ。

WHILE INKEY$ = " " : WEND

KEY 入力を待ちましょう。

CFLASH 0

KEY 入力が入ったらフラッシングを止めましょう。

120 **さて，いよいよ出題です。**

FOR i = 1 TO 11 : READ m$ (i) : NEXT i

11問の問題をすべて m$ の配列に読み込みましょう。

800 **DATA**

DATA	68	55	1
	出題部分	選択肢の部分	正解

DATA の中味は，こんな構成になっていますよ。もし皆さんが DATA をふやす場合は，必ずこのような構成にしてください。

130 **問題を5問つくりましょう。同じ問題は2度出ないよう**

にして……。

FOR i＝1 TO 11

まず，全部の問題の，

t（i）＝0

初期化を行います。全ての出題に0を書きます。

NEXT i

P＝0

これは，正解数の初期化。

FOR i＝1 TO 5

5問出題させましょう。

140　X＝INT（11＊RND（1））＋1

11題の中から無作為に，

XX＝t（X）

一度出題されたものは，

1F XX＜＞0 THEN 140 ELSE t（X）＝1

配列に1を入れておきましょう。t（X）が1の場合は，すでに出題済みという事で，他の問題を探しに行きます。

150　次の画面のために準備をしましょう！

CLS 0：CLS：PALET：WIDTH 40：COLOR 5

画面を消して，40文字モードに，色はシアンです。

160　何題目かを書きましょう。

POSITION 5，5：PATTERN－16，Ka$（37）

左上に“第”を書きましょう。

POSITION 23，5：PATTERN－16，Ka$（39＋i）

問題番号を書きましょう。

POSITION 41，5：PATTERN－16，Ka$（51）

“問”を書いて，でき上がり！

COLOR 4

色を緑に変えました。

170　いよいよ出題です。

LOCATE 0，5：PRINT LEFT$（m$（X），68）

問題を書きます。DATA の左から68個とりました。

LOCATE 3, 10 : PRINT MID$ (m$ (X), 69, 55)

選択肢を書きます。DATAの左から69個目から55個とり出して来ます。

180　さあ，正解はどれかなぁ……？

COLOR 7

色は白で，

LOCATE 3, 15 : PRINT "セイカイハナンバンデスカ"

とメッセージを書きました。

z$=INKEY$ (1) : 1 F 2$=" " THEN 180

ELSE z=VAL (z$)

解答をとり込むまで待ちましょう。解答は文字列で入ってくるので，VALで数字に換えましょう。

190　判定をしましょう！

IF z=VAL (RIGHT$ (m$ (X), 1)) THEN P=P+1

DATAの一番右から正解をとってきます。正解だったら正解数を1つふやしますよ。

LOCATE 23, 15 : CFLASH 1 : PRINT "セイカイ!!"

ELSE LOCATE 23, 15 : PRINT "マチガイ"

正解か間違えかのメッセージ。

200　次の問題にうつりましょう。

LOCATE 6, 23 : CFLASH 1 : PRINT "RETURN キーヲオシテクダサイ"

フラッシング・キャラクターでメッセージを。

WHILE INKEY$=" " : WEND : CFLASH 0

キー入力があったら，フラッシングをやめましょう。

NEXT i

さあ，次の問題にチャレンジ!!

210　結果は，どうだったかナ？

CLS 0 : CLS : PALET : COLOR 4

毎度おなじみの画面の初期設定です。

POSITION 100, 50 : PATTERN −16, Ka$ (39+P)

何問正解か，Pの値で決まります。

POSITION 118, 50 : PATTERN −16, Ka$ (51)
"問"を書きます。
POSITION 136, 50 : PATTERN −16, Ka$ (31)
"正"を書きます。
POSITION 154, 50 : PATTERN −16, Ka$ (52)
"解"を書きます。

220 COLOR 6 : LOCATE 15, 11 : PRINT "デシタ"
デシタを表示させます。

230 LOCATE 3, 15 : PRINT "モウイチドチョウセンシマスカ (y or n)"
と書きます。

240 Yes か No の判定
a$ = INKEY$: IF a$ = " " THEN 240 ELSE IF a$ = "y" THEN 130
y 以外全ての文字はうけつけません。

250 これで全てが終わりですよ。
CLS 0 : CLS : END
画面を消して，いよいよ END です。

さあ，楽しんでもらえました？　今まで勉強してきた事の集大成が，このプログラムです。だいたいの命令は入れてありますが，復習になりましたか，どうか。

正直言ってもう精一杯という感じなんですけど，もしもここはこう直した方がスムーズだというアイディアがありましたら，どんどん教えてくださいね。なんといっても，まだ始めたばかりの生徒ですから。またこのプログラムの内容に関しても，適切でないと思う方もいるでしょうが，これを参考にいろいろ考えてみてください。

またこのプログラムの内容に関しても，適切でないと思う方もいるでしょうが，これを参考にいろいろ考えて変えてみてください。

人それぞれに違う使い方があるはず。自分にとって BEST の使い方を考えてみるといいですネ。

```
10 '▓▓▓ ｾｶｲﾉｸﾆｸﾞﾆ (C) 1982,10 Hudsonsoft Y.Kudo Y.Shinagawa ▓▓▓
20 WIDTH 40:INIT:CLS4:MAXFILES0:T1$=HEXCHR$("AA0000550000"):T2$=HEXCHR$("00FFAA0
0FF55"):T3$=HEXCHR$("0000AA000055"):R$="Republic of "
30 GOSUB1530:ON A GOSUB 1580,1610,1640,1670,1700,1730:GOTO 30
40 C=7:GOSUB1270:X=75:Y=50:H=28:C=2:GOSUB1300:K=2:GOSUB1310:Q$="ﾆﾎﾝ":Q1$="Japan"
:Q2$="ﾄｳｷｮｳ":Q3$="378,000":Q4$="116,780":Q5$="1,019,317":TT=0:RETURN
50 GOSUB60:Q$="ｲｴﾒﾝ･ｱﾗﾌﾞｷｮｳﾜｺｸ":Q1$="Yemen Arab Republic":Q2$="ｻﾇｱ":Q3$="195":Q4
$="5,930":Q5$="2,394":TT=-6:RETURN
60 C=7:GOSUB1260:C1=2:C2=7:C3=0:GOSUB1290:X=75:Y=50:H=10:C=4:C1=7:GOSUB1430:RETU
RN
70 C=7:GOSUB1270:X=1:Y=8:X1=149:Y1=23:C=1:GOSUB1460:X=1:Y=76:X1=149:Y1=91:GOSUB1
460:X=75:Y=50:H=21:C=1:D=120:D1=210:D2=660:GOSUB1400:D1=270:GOSUB1400:Q$="ｲｽﾗｴﾙ"
:Q1$="State of Israel":Q2$="ﾃﾙｱﾋﾞﾌﾞ":Q3$="21":Q4$="3,920":Q5$="13,050":TT=-7:RET
URN
80 GOSUB60:X=35:Y=50:H=10:C=4:C1=7:GOSUB1430:X=115:GOSUB1430:Q$="ｲﾗｸｷｮｳﾜｺｸ":Q1$=
R$+"Iraq":Q2$="ﾊﾞｸﾞﾀﾞｯﾄﾞ":Q3$="435":Q4$="13,080":Q5$="30,366":TT=-6:RETURN
90 X=1:Y=1:X1=149:Y1=34:V$=T2$:GOSUB1490:Y=34:Y1=66:C=7:GOSUB1460:Y=66:Y1=99:V$=
T3$:GOSUB1490:X=75:Y=50:H=12:C=1:GOSUB1300:K=1:C=1:GOSUB1310:H=9:C=7:D=170:D1=0:
D2=3200:GOSUB1400:C=1:H=5:GOSUB1400
100 Q$="ｲﾝﾄﾞ":Q1$="India":Q2$="ﾆｭｰﾃﾞﾘｰ":Q3$="3,065":Q4$="683,810":Q5$="134,160":
TT=-4:RETURN
TT=-4:RETURN
110 C1=2:C2=7:GOSUB1340:Q$="ｲﾝﾄﾞﾈｼｱｷｮｳﾜｺｸ":Q1$=R$+"Indonesia":Q2$="ｼﾞｬｶﾙﾀ":Q3$="
1,904":Q4$="147,490":Q5$="52,873":TT=-2:RETURN
120 C=7:GOSUB1270:X=75:Y=50:C=2:H=24:GOSUB1300:K=2:GOSUB1310:CIRCLE(63,50),12,1,
1,180,360:CIRCLE(87,50),12,1,1,0,180:PAINT(77,50),1,1,7
130 COLOR0:LINE(35,9)-(50,18)-(35,39)-(20,30)-(35,9):LINE(35,91)-(50,82)-(35,61)
-(20,70)-(35,91):LINE(115,9)-(100,18)-(115,39)-(130,30)-(115,9):LINE(115,91)-(10
0,82)-(115,61)-(130,70)-(115,91)
140 PAINT(35,10),0,0:PAINT(35,90),0,0:PAINT(115,10),0,0:PAINT(115,90),0,0
150 LINE(38,11)-(23,32),PSET,7:LINE(41,12)-(26,33),PSET,7:LINE(44,15)-(29,36),PS
ET,7:LINE(47,16)-(32,37),PSET,7:PAINT(38,12),7,7:PAINT(44,16),7,7
160 LINE(112,11)-(127,32),PSET,7:LINE(109,12)-(124,33),PSET,7:LINE(106,15)-(121,
36),PSET,7:LINE(103,16)-(118,37),PSET,7:PAINT(112,12),7,7:PAINT(106,16),7,7
170 LINE(38,89)-(23,68),PSET,7:LINE(41,88)-(26,67),PSET,7:LINE(44,85)-(29,64),PS
ET,7:LINE(47,84)-(32,63),PSET,7:PAINT(38,88),7,7:PAINT(44,84),7,7
180 LINE(112,89)-(127,68),PSET,7:LINE(109,88)-(124,67),PSET,7:LINE(106,85)-(121,
64),PSET,7:LINE(103,84)-(118,63),PSET,7:PAINT(112,88),7,7:PAINT(106,84),7,7
190 LINE(35,79)-(38,76),PSET,7:LINE(32,74)-(35,72),PSET,7:PAINT(35,77),7,7:LINE(
```

```
121,82)-(106,73),PSET,7:LINE(124,78)-(109,69),PSET,7:PAINT(121,80),7,7:PAINT(109
,71),7,7:PAINT(115,77),7,7
200 LINE(121,18)-(118,20),PSET,7:LINE(109,25)-(106,26),PSET,7:LINE(124,23)-(121,
25),PSET,7:LINE(112,30)-(109,31),PSET,7:PAINT(121,19),7,7:PAINT(109,29),7,7:Q$="
ﾀﾞｲｶﾝﾐﾝｺｸ":Q1$=R$+"Korea":Q2$="ｿｳﾙ":Q3$="98":Q4$="38,200":Q5$="55,944":TT=0:RETU
RN

210 C=2:GOSUB1270:X=75:Y=50:H=5:C=6:D=5:D1=0:D2=180:GOSUB1500:X=65:H=4:GOSUB1500
:X=85:GOSUB1500
220 COLOR6:LINE(61,50)-(58,50)-(58,60)-(53,60)-(53,62)-(50,62)-(50,65)-(100,65)-
(100,62)-(97,62)-(97,60)-(92,60)-(92,50)-(89,50):X=75:Y=50:K=6:C=6:GOSUB1310
230 Q$="ﾐﾝｼｭｶﾝﾎﾞｼﾞｱ":Q1$="Democratic Kampuchea":Q2$="ﾌﾟﾉﾝﾍﾟﾝ":Q3$="181":Q1=1:Q4$
="ﾌﾒｲ":Q2=1:Q5$=Q4$:TT=-2:RETURN
240 C=7:GOSUB1260:C1=2:C2=7:C3=0:GOSUB1290:X=56:Y=50:H=10:C=4:C1=7:GOSUB1430:X=9
4:GOSUB1430:Q$="ｼﾘｱ･ｱﾗﾌﾞｷｮｳﾜｺｸ":Q1$="Syrian Arab Republic":Q2$="ﾀﾞﾏｽｶｽ":Q3$="185
":Q4$="9,620":Q5$="8,858":TT=-7:RETURN
250 C1=2:C2=7:GOSUB1340:X=35:Y=25:H=16:H1=16:C=7:X1=43:Y1=25:C1=2:GOSUB1510:C=7:
X=35:Y=23:H=2:D=144:D1=18:D2=900:GOSUB1400:X=55:GOSUB1400:X=45:Y=17:GOSUB1400
260 X=40:Y=33:GOSUB1400:X=50:GOSUB1400:Q$="ｼﾝｶﾞﾎﾟｰﾙ":Q1$=R$+"Singapore":Q2$="ｼﾝｶ
ﾞﾎﾟｰﾙ":Q=1:Q3$="600":Q4$="2,390":Q5$="9,192":TT=-2:RETURN
270 C1=7:C2=1:C3=7:GOSUB1290:X=1:Y=1:X1=149:Y1=16:C=2:GOSUB1460:Y=83:Y1=99:GOSUB
1460:Q$="ﾀｲｵｳｺｸ":Q1$="Kingdom of Thailand":Q2$="ﾊﾞﾝｺｯｸ":Q3$="514":Q4$="47,170":Q
5$="26,845":TT=-2:RETURN
280 C=2:GOSUB1270:X=26:Y=26:H=12:C=6:C1=2:GOSUB1430:X=50:Y=10:H=2:C=6:GOSUB1440:
X=60:Y=20:GOSUB1440:Y=35:GOSUB1440:X=50:Y=45:GOSUB1440
290 Q$="ﾁｭｳｶｼﾞﾝﾐﾝｷｮｳﾜｺｸ":Q1$="People's "+R$+"China":Q2$="ﾍﾟｷﾝ":Q3$="9,597":Q4$="
982,550":Q5$="250,770":TT=-1:RETURN
300 C=2:GOSUB1270:X=50:Y=50:H=26:C=7:X1=58:Y1=Y:H1=20:C1=2:GOSUB1510:X=83:C=7:H=
10:GOSUB1520:Q$="ﾄﾙｺｷｮｳﾜｺｸ":Q1$=R$+"Turkey":Q2$="ｱﾝｶﾗ":Q3$="781":Q4$="45,210":Q5
$="58,786":TT=-7:RETURN
310 C=4:GOSUB1270:X=1:Y=1:X1=36:Y1=99:C=7:GOSUB1460:X=92:Y=50:H=28:C=7:X1=97:Y1=
45:H1=24:C1=4:GOSUB1510
320 C=7:X=110:Y=35:H=10:C1=4:GOSUB1520:Q$="ﾊﾟｷｽﾀﾝｶｲｷｮｳｷｮｳﾜｺｸ":Q1$="Islamic "+R$+
"Pakistan":Q2$="ｲｽﾗﾏﾊﾞｰﾄﾞ":Q3$="804":Q4$="83,780":Q5$="23,298":TT=-4:RETURN
330 C=7:GOSUB1260:C=2:GOSUB1270:C=7:FORI=0TO90STEP10:X=30:Y=I:X1=50:Y1=I+5:GOSUB
1470:X=50:Y=I+5:X1=30:Y1=I+10:GOSUB1470:NEXT I:X=1:Y=1:H=7:C=7:K=7:GOSUB1310
340 Q$="ﾊﾞｰﾚﾝｺｸ":Q1$="State of Bahrain":Q2$="ﾏﾅｰﾏ":Q=1:Q3$="600":Q1=1:Q4$="360,0
00":Q5$="2,080":TT=-5:RETURN
350 C=7:GOSUB1270:FORI=1TO85STEP14:LINE(1,I)-(149,I+7),PSET,2,BF:NEXT:X=1:Y=1:X1
=75:Y1=56:C=1:GOSUB1460
360 X=30:Y=28:H=20:C=6:X1=38:Y1=28:H1=20:C1=1:GOSUB1510:X=38:H=16:C=1:GOSUB1300:
K=1:GOSUB1310:X=55:C=6:D=128:D1=0:D2=2000
370 FORH=13TO1STEP-1:GOSUB1400:NEXT:Q$="ﾏﾚｰｼｱ":Q1$="Malaysia":Q2$="ｸｱﾗﾙﾝﾌﾟｰﾙ":Q3
$="330":Q4$="13,440":Q5$="17,947":TT=-2:RETURN
380 C=2:GOSUB1270:X=20:Y=19:X1=130:Y1=80:C=4:GOSUB1460:X=85:Y=50:H=26:C=7:X1=95:
Y1=50:H1=26:C1=4:GOSUB1510:Q$="ﾓﾙｼﾞﾌﾞｷｮｳﾜｺｸ":Q1$=R$+"Maldives":Q2$="ﾏﾚ":Q=1:Q3$=
"300":Q1=1:Q4$="150,000":Q5$="23":TT=-4:RETURN
390 C=7:GOSUB1260:C1=0:C2=7:C3=4:GOSUB1290:C=2:C1=2:GOSUB1320
400 X=27:Y=50:C=7:D=103:D1=0:D2=721:FORH=8TO1STEP-1:GOSUB1400:NEXT:Q$="ﾖﾙﾀﾞﾝﾊｼﾐﾃ
ｵｳｺｸ":Q1$="Hashemite Kingdom of Jordan":Q2$="ｱﾝﾏﾝ":Q3$="98":Q4$="3,190":Q5$="3,
658":TT=-7:RETURN
```

```
410 GOSUB1250:X=66:Y=150:C=7:D=154:D1=0:D2=1081:FORH=16TO1STEP-1:GOSUB1400:NEXT:
X=235:Y=30:H=8:GOSUB1400:X=180:Y=80:GOSUB1400:X=265:Y=75:GOSUB1400
420 X=225:Y=165:GOSUB1400:H=3:X=250:Y=111:GOSUB1400:Q$="ｵｰｽﾄﾗﾘｱ":Q1$="Australia"
:Q2$="ｷｬﾝﾍﾞﾗ":Q3$="7,687":Q4$="14,620":Q5$="130,416":TT=1:RETURN

430 GOSUB1360:LINE(1,1)-(312,199),PSET,5,BF:GOSUB770:C=6:C1=6:H=12:X=275:Y=29:GO
SUB1430:X=245:Y=55:GOSUB1520:X=215:Y=70:GOSUB1520
440 X=280:Y=100:GOSUB1430:X=245:Y=125:GOSUB1430:X=195:Y=130:GOSUB1450:X=225:Y=15
0:GOSUB1450:X=195:Y=175:GOSUB1450:X=160:Y=185:GOSUB1430:Q$="ﾂﾊﾞﾙ":Q1$="Tuvalu":Q
2$="ﾌﾅﾌﾁ":Q=1:Q3$="30":Q1=1:Q4$="10,000":Q2=1:Q5$="ﾌﾒｲ":TT=3:RETURN
450 C=2:GOSUB1270:X=1:Y=1:X1=75:Y1=50:C=7:GOSUB1460:X=30:Y=5:X1=44:Y1=45:C=2:GOS
UB1460:X=17:Y=18:X1=57:Y1=32:GOSUB1460:Q$="ﾄﾝｶﾞｵｳｺｸ":Q1$="Kingdom of Tonga":Q2$=
"ﾇｸｱﾛﾌｧ":Q=1:Q3$="700":Q1=1:Q4$="100,000":Q2=1:Q5$="$38,700,000":TT=3:RETURN
460 GOSUB1250:H=16:C=3:C1=3:X=225:Y=35:GOSUB1430:X=255:Y=75:H=12:GOSUB1430:H=16:
X=195:Y=75:GOSUB1430:X=225:Y=155:GOSUB1430:Q$="ﾆｭｰｼﾞｰﾗﾝﾄﾞ":Q1$="New Zealand":Q2$
="ｳｪﾘﾝﾄﾝ":Q3$="269":Q4$="3,100":Q5$="18,976"TT=3:RETURN
470 X=1:Y=1:X1=75:Y1=99:C=4:GOSUB1460:X=76:Y=1:X1=149:C=7:GOSUB1460:X=75:Y=50:H=
28:C=2:X1=82:Y1=50:H1=22:C1=7:GOSUB1510
480 LINE(75,29)-(75,71),PSET,4:PAINT(73,55),4,2,4:X=87:Y=50:H=13:C=2:GOSUB1450:Q
$="ｱﾙｼﾞｪﾘｱﾐﾝｼｭｼﾞﾝﾐﾝ    ｷｮｳﾜｺｸ":Q1$="Democratic "+R$+"Algeria":Q2$="ｱﾙｼﾞｪ":Q3$="2
,382":Q4$="18,190":Q5$="28,938":TT=-9:RETURN
490 C1=4:C2=6:C3=2:GOSUB1290:Q$="ｼｬｶｲｼｭｷﾞｴﾁｵﾋﾟｱ":Q1$="Socialist Etiopia":Q2$="ｱｼ
ﾞｽｱﾍﾞﾊﾞ":Q3$="1,222":Q4$="31,070":Q5$="4,017":TT=-6:RETURN
500 C1=4:C2=2:C3=6:GOSUB1280:X=75:Y=50:H=10:C=6:C1=2:GOSUB1430:Q$="ｶﾒﾙｰﾝﾚﾝｺﾞｳｷｮｳ
ﾜｺｸ":Q1$="United "+R$+"Cameroon":Q2$="ﾔｳﾝﾃﾞ":Q3$="475":Q4$="8,500":Q5$="4,592":T
T=-8:RETURN
510 C1=2:C2=6:C3=4:GOSUB1280:Q$="ｷﾞﾆｱｼﾞﾝﾐﾝｶｸﾒｲｷｮｳﾜｺｸ":Q1$="People's Revolutionar
y of Guinea":Q2$="ｺﾅｸﾘ":Q3$="246":Q4$="5,270":Q5$="1,540":TT=-9:RETURN
520 C=7:GOSUB1260:C1=0:C2=2:C3=4:GOSUB1290:X=1:Y=31:X1=149:Y1=34:C=7:GOSUB1460:Y
=66:Y1=69:GOSUB1460:CIRCLE(75,50),15,2,2:PAINT(75,50),2,7,2:X=75:Y=25:K=2:C=2:GO
SUB1310:Y=75:GOSUB1310:CIRCLE(63,50),4,0,4:X=63:Y=50:K=0:C=0:GOSUB1310
530 CIRCLE(87,50),4,0,4:X=87:GOSUB1310:CIRCLE(75,35),2,7,4:CIRCLE(75,65),2,7,4:X
=75:Y=35:K=7:C=7:GOSUB1310:Y=65:GOSUB1310:COLOR2:LINE(75,20)-(75,80):CIRCLE(75,5
0),2,7,2:Y=50:GOSUB1310
540 COLOR7:LINE(60,70)-(55,80):LINE(90,70)-(95,80):COLOR 7:LINE(55,13)-(65,28):L
INE(95,13)-(85,28):Q$="ｹﾆｱｷｮｳﾜｺｸ":Q1$=R$+"Kenya":Q2$="ﾅｲﾛﾋﾞ":Q3$="583":Q4$="16,4
00":Q5$="5,814":TT=-6:RETURN
550 C1=2:C2=2:C3=4:GOSUB 1290:X=30:Y=25:H=17:C=7:X1=35:Y1=30:H1=17:C1=2:GOSUB151
0
560 POLY(30,25),3,7,144,0,720:POLY(45,27),3,7,144,0,720:POLY(33,35),3,7,144,0,72
0:POLY(48,38),3,7,144,0,720:Q$="ｺﾓﾛﾚﾝﾎﾟｳｶｲｷｮｳｷｮｳﾜｺｸ":Q1$="Federal & Islamic "+R$
+"Comoros":Q2$="ﾓﾛﾆ":Q3$="2":Q1=1:Q4$="340,000":Q5$="90":TT=-6:RETURN
570 C1=4:C2=6:C3=4:GOSUB1290:C=2:C1=2:GOSUB1330:X=70:Y=50:H=10:C=0:C1=6:GOSUB143
0:X=115:GOSUB1430:Q$="ｻﾝﾄﾒﾌﾟﾘﾝｼﾍﾟﾐﾝｼｭｷｮｳﾜ ｺｸ":Q1$="Sao Tome and Principe":Q2$="ｻ
ﾝﾄﾒ":Q3$="1":Q1=1:Q4$="90":Q5$="52":TT=-8:RETURN
580 C1=4:C2=7:C3=1:GOSUB1290:Q$="ｼｴﾗﾚｵﾈｷｮｳﾜｺｸ":Q1$=R$+"Sierra Leone":Q2$="ﾌﾘｰﾀｳﾝ
":Q3$="72":Q4$="3,470":Q5$="850":TT=-9:RETURN
590 C=7:GOSUB1260:C1=2:C2=7:C3=0:GOSUB1290:C=4:C1=4:GOSUB1320:Q$="ｽｰﾀﾞﾝﾐﾝｼｭｷｮｳﾜｺ
ｸ":Q1$="Democratic "+R$+"the Sudan":Q2$="ﾊﾙﾂｰﾑ":Q3$="2,506":Q4$="18,690":Q5$="6,
623":TT=-7:RETURN
```

```
600 C=7:GOSUB1260:C1=2:C2=2:C3=4:GOSUB1290:FORJ=0TO15STEP15:X0=51+J:FORI=1TO150:
I0=I-1:X1=-COS(PAI(4)/150*I)*5+56+J:LINE(I0,X0)-(I,X1),PSET,7:X0=X1:NEXTI,J:PAIN
T(2,52),7,7

610 Q$="ｾｲｼｪﾙｷｮｳﾜｺｸ":Q1$=R$+"Seychelles":Q2$="ﾋﾞｸﾄﾘｱ":Q=1:Q3$="4000":Q1=1:Q4$="6
0":Q5$="90":TT=-5:RETURN
620 C1=4:C2=6:C3=2:GOSUB1280:X=75:Y=50:H=10:C=4:C1=6:GOSUB1430:Q$="ｾﾈｶﾞﾙｷｮｳﾜｺｸ":
Q1$=R$+"Senegal":Q2$="ﾀﾞｶｰﾙ":Q3$="197":Q4$="5,660":Q5$="2,365":TT=-9:RETURN
630 C=5:GOSUB1270:X=75:Y=50:H=15:C=7:C1=5:GOSUB1430:Q$="ｿﾏﾘｱﾐﾝｼｭｷｮｳﾜｺｸ":Q1$="Som
ali Democratic Republic":Q2$="ﾓｶﾞｼﾞｼｵ":Q3$="638":Q4$="3,650":Q2=1:Q5$="ﾌﾒｲ":TT=-
6:RETURN
640 C=7:GOSUB1260:LINE(110,1)-(1,75),PSET,6:LINE(120,1)-(1,82),PSET,6:LINE(149,2
0)-(28,99),PSET,6:LINE(149,27)-(38,99),PSET,6:PAINT(2,2),4,6,7:PAINT(148,98),1,6
,7:PAINT(112,2),6,6,7:PAINT(31,98),6,6,7
650 Q$="ﾀﾝｻﾞﾆｱﾚﾝｺﾞｳｷｮｳﾜｺｸ":Q1$="United "+R$+"of Tanzania":Q2$="ﾀﾞﾙｴｽｻﾗｰﾑ":Q3$="9
45":Q4$="17,540":Q5$="4,700":TT=-6:RETURN
660 C1=1:C2=7:C3=4:C4=6:GOSUB1380:LINE(62,1)-(88,99),PSET,2,BF:X=30:Y=12:H=10:C=
6:C1=1:GOSUB1430:Q$="ﾁｭｳｵｳｱﾌﾘｶｷｮｳﾜｺｸ":Q1$="Central African Republic":Q2$="ﾊﾞﾝｷﾞ"
:Q3$="623":Q4$="3,050":Q5$="580":TT=-8:RETURN
670 C=2:GOSUB1270:X=75:Y=50:H=33:C=7:GOSUB1300:K=7:GOSUB1310:H=22:C=2:X1=85:Y1=5
0:C1=7:H1=22:GOSUB1510:X=85:Y=50:H=13:C=2:C1=7:GOSUB1430:Q$="ﾁｭﾆｼﾞｱｷｮｳﾜｺｸ":Q1$=R
$+"Tunisia":Q2$="ﾁｭﾆｽ":Q3$="165":Q4$="6,360":Q5$="6,944":TT=-8:RETURN
680 C1=4:C2=6:C3=4:C4=6:C5=4:GOSUB1390:LINE(1,1)-(60,60),PSET,2,BF:X=30:Y=30:H=1
2:C=7:C1=2:GOSUB1430:Q$="ﾄｰｺﾞｷｮｳﾜｺｸ":Q1$=R$+"Togo":Q2$="ﾛﾒ":Q3$="57":Q4$="2,470"
:Q5$="840":TT=-8:RETURN
690 C=4:GOSUB1270:X=25:Y=25:H=10:C=2:C1=4:GOSUB1430:Q$="ﾍﾞﾅﾝｼﾞﾝﾐﾝｷｮｳﾜｺｸ":Q1$="Pe
ople's "+R$+"Benin":Q2$="ﾎﾟﾙﾄﾉﾎﾞ":Q3$="113":Q4$="3,570":Q5$="850":TT=-8:RETURN
700 C=4:GOSUB1270:X=75:Y=25:H=40:C=6:X1=75:Y1=15:H1=40:C1=4:GOSUB1510:C=0:GOSUB1
260:H=12:C=6:C1=4:GOSUB1430:Q$="ﾓｰﾘﾀﾆｱｶｲｷｮｳｷｮｳﾜｺｸ":Q1$="Islamic "+R$+"Mauritania
":Q2$="ﾇｱｸｼｮｯﾄ":Q3$="1,031":Q4$="1,590":Q5$="512":TT=-9:RETURN
710 C=2:GOSUB1270:POLY(75,50),15,4,144,18,900:Q$="ﾓﾛｯｺｵｳｺｸ":Q1$="Kingdom of Moro
cco":Q2$="ﾗﾊﾞﾄ":Q3$="447":Q4$="20,130":Q5$="14,460":RETURN
720 C=4:GOSUB1270:Q$="ｼｬｶｲｼｭｷﾞﾘﾋﾞｱ･ｱﾗﾌﾞｺｸ":Q1$="Libyan Arab Jamahiriya":Q3$="1,7
60":Q2$="ﾄﾘﾎﾟﾘ":Q4$="2,980":Q5$="23,693":TT=-7:RETURN
730 C=7:GOSUB1270:X=1:X1=149:C=2:FORI=1TO99STEP18:Y=I:Y1=I+9:GOSUB1460:NEXT:LINE
(1,1)-(45,46),PSET,1,BF:X=22:Y=22:H=10:C=7:C1=1:GOSUB1430:Q$="ﾘﾍﾞﾘｱｷｮｳﾜｺｸ":Q1$=R
$+"Liberia":Q2$="ﾓﾝﾛﾋﾞｱ":Q3$="111":Q4$="1,870":Q5$="900":TT=-9:RETURN
740 C=1:GOSUB1270:C=7:GOSUB1410:C=2:GOSUB1420:Q$="ｱｲｽﾗﾝﾄﾞｷｮｳﾜｺｸ":Q1$=R$+"Iceland
":Q2$="ﾚｲｷｬﾋﾞｸ":Q3$="103":Q1=1:Q4$="230,000":Q5$="2,370":TT=-10:RETURN
750 C1=4:C2=7:C3=6:GOSUB1280:X=100:Y=1:X1=149:Y1=99:V$=T2$:GOSUB1490:Q$="ｱｲﾙﾗﾝﾄﾞ
":Q1$="Ireland":Q2$="ﾀﾞﾌﾞﾘﾝ":Q3$="70":Q4$="3,370":Q5$="13,893":TT=-9:RETURN
760 GOSUB770:Q$="ｲｷﾞﾘｽ(ｸﾞﾚｰﾄﾌﾞﾘﾃﾝ)":Q1$="United Kingdom of Great Britain":Q2$="ﾛ
ﾝﾄﾞﾝ":Q3$="244":Q4$="55,950":Q5$="353,288":TT=-9:RETURN
770 C=7:GOSUB1260:C=1:V$=T1$:GOSUB1370:LINE(1,34)-(149,66),PSET,7,BF:LINE(59,1)-
(91,99),PSET,7,BF
780 LINE(1,40)-(149,60),PSET,2,BF:LINE(65,1)-(85,99),PSET,2,BF
790 COLOR 7:LINE(15,1)-(60,30):LINE(34,34)-(1,12):PAINT(5,7),7,7
800 LINE(135,0)-(91,31):LINE(149,12)-(115,34):PAINT(125,20),7,7
810 LINE(1,90)-(38,66):LINE(60,72)-(17,100):PAINT(10,90),7,7
820 LINE(113,66)-(149,90):LINE(91,71)-(133,100):PAINT(110,80),7,7
830 COLOR 2:LINE(1,1)-(1,10)-(37,33)-(51,33)-(1,1):PAINT(5,7),2,2
```

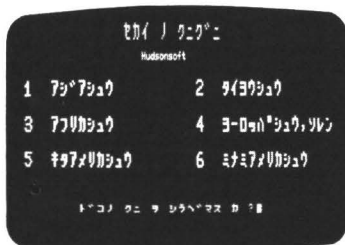
```
1070 FORY=10TO40STEP10:FORX=13TO57STEP11:GOSUB1400:NEXTX,Y:Q$="ｱﾒﾘｶｶﾞｯｼｭｳｺｸ":Q1$
="United States of America":Q2$="ﾜｼﾝﾄﾝ":Q3$="9,363":Q4$="226,500":Q5$="2,376,868
":TT=-14:RETURN
1080 C=7:GOSUB1270:LINE(1,1)-(37,99),PSET,2,BF:LINE(113,1)-(149,99),PSET,2,BF:CO
LOR2:LINE(73,85)-(77,85)-(77,70)-(97,73)-(95,68)-(110,50)-(105,48)-(110,35)-(95,
40)-(93,35)-(87,43)-(90,20)-(82,23)-(75,10)-(68,23)-(60,20)-(63,45)
1090 LINE(63,45)-(57,35)-(55,40)-(40,35)-(45,48)-(40,50)-(55,68)-(53,73)-(73,70)
-(73,85):PAINT(75,50),2,2:Q$="ｶﾅﾀﾞ":Q1$="Canada":Q2$="ｵﾀﾜ":Q3$="9,976":Q4$="24,0
90":Q5$="228,468":TT=-14:RETURN
1100 C=7:GOSUB1260:LINE(15,1)-(149,90),PSET,6:LINE(1,10)-(135,99),PSET,6:LINE(13
5,1)-(1,90),PSET,6:LINE(149,10)-(15,99),PSET,6:LINE(65,45)-(85,55),PSET,0,B
1110 PAINT(75,50),6,6,7:PAINT(75,1),4,6,7:PAINT(75,99),4,6,7:Q$="ｼﾞｬﾏｲｶ":Q1$="Ja
maica":Q2$="ｷﾝｸﾞｽﾄﾝ":Q3$="11":Q4$="2,190":Q5$="2,772":TT=-14:RETURN
1120 C=1:GOSUB1270:COLOR7:LINE(40,85)-(75,8)-(110,85)-(40,85):PAINT(75,50),7,7:C
OLOR 0:LINE(50,85)-(75,26)-(100,85)-(50,85):PAINT(75,50),0,0:COLOR6:LINE(40,85)-
(75,60)-(110,85)-(40,85):PAINT(75,70),6,6
1130 Q$="ｾﾝﾄﾙｼｱ":Q1$="Saint Lucia":Q2$="ｶｽﾄﾘｰｽﾞ":Q=1:Q3$="600":Q1=1:Q4$="120,00-
":Q5$="87":TT=-14:RETURN
1140 C=7:GOSUB1270:LINE(76,1)-(149,50),PSET,2,BF:LINE(1,51)-(75,99),PSET,1,BF:X=
37:Y=25:H=12:C=1:C1=7:GOSUB1430:X=113:Y=75:C=2:GOSUB1430:Q$="ﾊﾟﾅﾏｷｮｳﾜｺｸ":Q1$=R$+
"Panama":Q2$="ﾊﾟﾅﾏ":Q3$="76":Q4$="1,840":Q5$="2,520":TT=-14:RETURN
1150 C=7:GOSUB1260:C1=5:C2=6:C3=5:GOSUB1290:C=0:C1=0:GOSUB1320:Q$="ﾊﾞﾊﾏｺｸ":Q1$="
Commonwealth of Bahamas":Q2$="ﾅｯｿｰ":Q3$="14":Q1=1:Q4$="240,000":Q5$="640":TT=-14
:RETURN
1160 C1=5:C2=7:C3=5:GOSUB1290:X=75:Y=50:H=6:C=5:C1=7:GOSUB1430:X=63:Y=40:GOSUB14
30:Y=60:GOSUB1430:X=87:Y=40:GOSUB1430:Y=60:GOSUB1430:Q$="ﾎﾝｼﾞｭﾗｽｷｮｳﾜｺｸ":Q1$=R$+"
Honduras":Q2$="ﾃｸﾞｼｶﾞﾙﾊﾟ":Q3$="112":Q4$="3,690":Q5$="1,908":TT=-15:RETURN
1170 C=7:GOSUB1260:V$=T3$:GOSUB1370:COLOR7:LINE(1,1)-(149,50)-(1,99)-(1,1):PAINT
(147,50),7,7:COLOR6:LINE(1,5)-(137,50)-(1,95)-(1,5):PAINT(135,50),6,6
1180 C=0:C1=0:GOSUB1320:COLOR2:LINE(1,5)-(68,50)-(1,95)-(1,5):PAINT(3,7),2,2:Q$=
"ｶﾞｲｱﾅｷｮｳﾄﾞｳｷｮｳﾜｺｸ":Q1$="The Cooperative "+R$+"Guyana":Q2$="ｼﾞｮｰｼﾞﾀｳﾝ":Q3$="215"
:Q1=1:Q4$="880,000":Q5$="480":TT=-13:RETURN
1190 C1=4:C2=2:C3=4:GOSUB1290:LINE(1,24)-(149,34),PSET,7,BF:LINE(1,66)-(149,76),
PSET,7,BF
1200 X=75:Y=50:H=13:C=6:C1=2:GOSUB1430:Q$="ｽﾘﾅﾑｷｮｳﾜｺｸ":Q1$=R$+"Surinam":Q2$="ﾊﾟﾗ
ﾏﾘﾎﾞ":Q3$="163":Q1=1:Q4$="390,000":Q5$="950":TT=-13:RETURN
1210 C1=7:C2=2:GOSUB1340:LINE(1,1)-(50,49),PSET,1,BF:X=25:Y=25:H=12:C=7:C1=1:GOS
UB1430:Q$="ﾁﾘｷｮｳﾜｺｸ":Q1$=R$+"Chile":Q2$="ｻﾝﾁｱｺﾞ":Q3$="757":Q4$="11,100":Q5$="18,
421":TT=-13:RETURN

1220 V$=T3$:GOSUB1370:COLOR6:LINE(75,10)-(10,50)-(75,90)-(140,50)-(75,10):X=75:Y
=50:K=6:C=6:GOSUB1310:H=22:C=1:GOSUB1300:K=1:GOSUB1310
1230 CIRCLE(60,99),60,6,1,45,120:CIRCLE(55,104),60,6,1,45,120:X=85:Y=40:H=2:C=6:
D=144:D1=0:D2=720:GOSUB1400
1240 H=1:X=70:Y=50:GOSUB1400:Y=55:GOSUB1400:X=75:Y=55:GOSUB1400:Y=60:GOSUB1400:Q
$="ﾌﾞﾗｼﾞﾙﾚﾝﾎﾟｳｷｮｳﾜｺｸ":Q1$="Federative "+R$+"Brazil":Q2$="ﾌﾞﾗｼﾞﾘｱ":Q3$="8,512":Q4
$="123,033":Q5$="207,370":TT=-12:RETURN
1250 GOSUB1360:LINE(0,0)-(312,199),PSET,1,BF,T1$:GOSUB770:LINE(1,1)-(150,100),PS
ET,1,B:RETURN:'ｲｷﾞﾘｽ1/4
1260 LINE(0,0)-(150,100),PSET,C,B:RETURN:'ﾜｸ
1270 LINE(1,1)-(149,99),PSET,C,BF:RETURN:'ﾇﾘﾂﾌﾞｼ
```

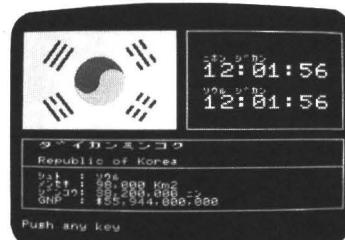
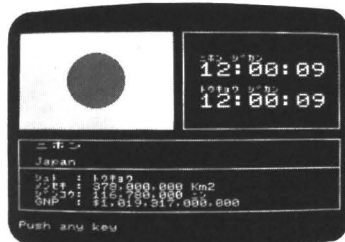
```
840 LINE(138,1)-(149,1)-(100,33)-(92,33)-(92,32)-(138,1):PAINT(130,10),2,2
850 LINE(111,67)-(149,92)-(149,99)-(101,67)-(111,67):PAINT(140,90),2,2
860 LINE(49,67)-(58,67)-(58,69)-(15,99)-(1,99)-(49,67):PAINT(30,85),2,2:RETURN

870 C1=4:C2=7:C3=2:GOSUB1280:Q$="ｲﾀﾘｱｷｮｳﾜｺｸ":Q1$=R$+"Italy":Q2$="ﾛｰﾏ":Q3$="301":
Q4$="57,040":Q5$="298,200":TT=-8:RETURN
880 C1=2:C2=7:C3=2:GOSUB1290:Q$="ｵｰｽﾄﾘｱｷｮｳﾜｺｸ":Q1$=R$+"Austria":Q2$="ｳｨｰﾝ":Q3$="
84":Q4$="7,510":Q5$="64,725":TT=-8:RETURN
890 C1=2:C2=7:C3=5:GOSUB1290:Q$="ｵﾗﾝﾀﾞｵｳｺｸ":Q1$="Kingdom of the Netherlands":Q2$
="ｱﾑｽﾃﾙﾀﾞﾑ":Q3$="41":Q4$="14,140":Q5$="143,220":TT=-8:RETURN
900 C=7:GOSUB1270:FORI=1TO99STEP22:LINE(1,I)-(149,I+11),PSET,1,BF:NEXT:LINE(1,1)
-(55,55),PSET,1,BF
910 LINE(1,23)-(55,34),PSET,7,BF:LINE(22,1)-(32,56),PSET,7,BF:Q$="ｷﾞﾘｼｱｷｮｳﾜｺｸ":Q
1$="Hellenic Republic":Q2$="ｱﾃﾈ":Q3$="132":Q4$="9,600":Q5$="36,828":TT=-7:RETURN

920 C=2:GOSUB1270:LINE(40,40)-(110,60),PSET,7,BF:LINE(65,15)-(85,85),PSET,7,BF:Q
$="ｽｲｽﾚﾝﾎﾟｳ":Q1$="Swiss Confederation":Q2$="ﾍﾞﾙﾝ":Q3$="41":Q4$="6,370":Q5$="90,4
80":TT=-8:RETURN
930 C=1:GOSUB1270:C=6:GOSUB1410:Q$="ｽｳｪｰﾃﾞﾝｵｳｺｸ":Q1$="Kingdom of Sweden":Q2$="ｽﾄ
ｯｸﾎﾙﾑ":Q3$="450":Q4$="8,310":Q5$="99,019":TT=-8:RETURN
940 C1=7:C2=2:GOSUB1340:C=1:C1=1:GOSUB1320:Q$="ﾁｪｺｽﾛﾊﾞｷｱｼｬｶｲｼｭｷﾞ   ｷｮｳﾜｺｸ":Q1$="
Czechoslovak Socialist Republic":Q2$="ﾌﾟﾗﾊ":Q3$="128":Q4$="15,320":Q5$="80,408":
TT=-8:RETURN
950 C=2:GOSUB1270:C=7:GOSUB1420:Q$="ﾃﾞﾝﾏｰｸｵｳｺｸ":Q1$="Kingdom of Denmark":Q2$="ｺﾍ
ﾟﾝﾊｰｹﾞﾝ":Q3$="43":Q4$="5,120":Q5$="60,690":TT=-8:RETURN
960 C=7:GOSUB1260:C1=0:C2=2:C3=6:GOSUB1290:X=1:Y=66:X1=149:Y1=99:V$=T2$:GOSUB149
0:Q$="ﾄﾞｲﾂﾚﾝﾎﾟｳｷｮｳﾜｺｸ":Q1$="Federal "+R$+"Germany":Q2$="ﾎﾞﾝ":Q3$="249":Q4$="61,5
60":Q5$="717,876":TT=-8:RETURN
970 C=2:GOSUB1270:C=7:GOSUB1410:C=1:GOSUB1420:Q$="ﾉﾙｳｪｰｵｳｺｸ":Q1$="Kingdom of Nor
way":Q2$="ｵｽﾛ":Q3$="324":Q4$="4,090":Q5$="43,870":TT=-8:RETURN
980 C=7:GOSUB1270:C=5:GOSUB1410:Q$="ﾌｨﾝﾗﾝﾄﾞｷｮｳﾜｺｸ":Q1$="Republic of Finland":Q2$
="ﾍﾙｼﾝｷ":Q3$="337":Q4$="4,780":Q5$="39,168":TT=-7:RETURN
990 C1=1:C2=7:C3=2:GOSUB1280:Q$="ﾌﾗﾝｽｷｮｳﾜｺｸ":Q1$="French Republic":Q2$="ﾊﾟﾘ":Q3$
="547":Q4$="53,710":Q5$="531,330":TT=-8:RETURN
1000 C=7:GOSUB1260:C1=0:C2=6:C3=2:GOSUB1280:Q$="ﾍﾞﾙｷﾞｰｵｳｺｸ":Q1$="Kingdom of Belg
ium":Q2$="ﾌﾞﾘｭｯｾﾙ":Q3$="31":Q4$="9,860":Q5$="107,016":TT=-8:RETURN
1010 C1=1:C2=7:C3=2:GOSUB1290:X=75:Y=50:H=28:C=6:GOSUB1440:H=26:C=2:C1=7:GOSUB 1
430:Q$="ﾕｰｺﾞｽﾗﾋﾞｱｼｬｶｲｼｭｷﾞ   ﾚﾝﾎﾟｳｷｮｳﾜｺｸ":Q1$="SocialistFederal"+R$+"Yugoslavia":
Q2$="ﾍﾞｵｸﾞﾗｰﾄﾞ":Q3$="256":Q4$="22,340":Q5$="53,703":TT=-8:RETURN
1020 C=2:GOSUB1270:X=25:Y=20:C=6:H=12:X1=22:Y1=17:H1=12:C1=2:GOSUB1510:COLOR6:LI
NE(12,17)-(18,11)-(25,11)-(14,21)-(12,17):LINE(15,28)-(18,30)-(11,34)-(8,30)-(15
,28):PAINT(14,30),6,6
1030 PAINT(16,15),6,6:COLOR6:LINE(17,13)-(37,33)-(35,35)-(15,15)-(17,13):PAINT(1
9,18),6,6:PAINT(35,34),6,6:POLY(33,6),4,6,144,18,900
1040 Q$="ｿﾋﾞｴﾄｼｬｶｲｼｭｷﾞｷｮｳﾜｺｸ ﾚﾝﾎﾟｳ":Q1$="Union of Soviet Socialist Republics":Q2
$="ﾓｽｸﾜ":Q3$="22,402":Q4$="267,700":Q5$="1,085":TT=-6:RETURN

1050 LINE(1,1)-(149,92),PSET,7,BF:FOR I=1 TO 92 STEP 14:LINE(1,I)-(149,I+7),PSET
,2,BF:NEXT I:LINE(1,1)-(70,50),PSET,1,BF
1060 H=2:C=7:D=144:D1=18:D2=900:FORY=5TO50STEP10:FORX=8TO65STEP11:GOSUB1400:NEXT
X,Y
```

```
1280 LINE(1,1)-(50,99),PSET,C1,BF:LINE(50,1)-(100,99),PSET,C2,BF:LINE(100,1)-(14
9,99),PSET,C3,BF:RETURN:'ﾀﾃ 3 ﾄｳﾌﾞﾝ
1290 LINE(1,1)-(149,34),PSET,C1,BF:LINE(1,34)-(149,66),PSET,C2,BF:LINE(1,66)-(14
9,99),PSET,C3,BF:RETURN:'ﾖｺ 3 ﾄｳﾌﾞﾝ
1300 CIRCLE(X,Y),H,C:RETURN:'ﾏﾙ
1310 PAINT(X,Y),K,C:RETURN:'ﾇﾘﾂﾌﾞｼ
1320 COLORC:LINE(1,1)-(75,50)-(1,99)-(1,1):PAINT(40,50),C,C1:RETURN:'ﾋﾀﾞﾘ3ｶｸ1/2
1330 COLORC:LINE(1,1)-(50,50)-(1,99)-(1,1):PAINT(40,50),C,C1:RETURN:'ﾋﾀﾞﾘ3ｶｸ1/4
1340 LINE(1,1)-(149,50),PSET,C1,BF:LINE(1,50)-(149,99),PSET,C2,BF:RETURN:'ﾖｺ1/2
1350 LINE(1,1)-(149,25),PSET,C1,BF:LINE(1,25)-(149,50),PSET,C2,BF:LINE(1,50)-(14
9,75),PSET,C3,BF:LINE(1,75)-(149,99),PSET,C4,BF:RETURN:'ﾖｺ1/4
1360 WINDOW(2,1)-(319,199),(0,1)-(639,399):RETURN:'ｲｷﾞﾘｽ ｳｲﾝﾄﾞ
1370 LINE(1,1)-(148,99),PSET,C,BF,V$:RETURN:'ﾇﾘﾂﾌﾞｼ
1380 LINE(1,1)-(149,25),PSET,C1,BF:LINE(1,26)-(149,50),PSET,C2,BF:LINE(1,51)-(14
9,75),PSET,C3,BF:LINE(1,76)-(149,99),PSET,C4,BF:RETURN:'ﾖｺ1/4
1390 LINE(1,1)-(149,20),PSET,C1,BF:LINE(1,21)-(149,40),PSET,C2,BF:LINE(1,41)-(14
9,60),PSET,C3,BF:LINE(1,61)-(149,80),PSET,C4,BF:LINE(1,81)-(149,99),PSET,C5,BF:R
ETURN:'ﾖｺ1/5
1400 POLY(X,Y),H,C,D,D1,D2:RETURN:'ﾎﾟｰﾘｰ
1410 LINE(1,35)-(149,65),PSET,C,BF:LINE(35,1)-(65,99),PSET,C,BF:RETURN:'+ﾌﾄｲ
1420 LINE(1,42)-(149,58),PSET,C,BF:LINE(42,1)-(58,99),PSET,C,BF:RETURN:'+ﾎｿｲ
1430 POLY(X,Y),H,C,144,18,900:POLY(X,Y),H/2,C1,144,18,900:PAINT(X,Y),C,C:RETURN:
'ｳｴ ﾎｼ
1440 POLY(X,Y),H,C,144,18,900:RETURN:'ｳｴﾎｼ1ﾂ
1450 POLY(X,Y),H,C,144,0,720:POLY(X,Y),H/2,C1,144,0,720:PAINT(X,Y),C,C:RETURN:'ﾐ
ｷﾞﾎｼ
1460 LINE(X,Y)-(X1,Y1),PSET,C,BF:RETURN:'ﾎﾞｯｸｽﾌﾙ
1470 LINE(X,Y)-(X1,Y1),PSET,C:RETURN:'ﾗｲﾝ
1480 LINE(X,Y)-(X1,Y1),PSET,C,B:RETURN:'ﾎﾞｯｸｽ
1490 LINE(X,Y)-(X1,Y1),PSET,C,BF,V$:RETURN:'ﾀｲﾘﾝｸﾞ
1500 CIRCLE(X,Y),H,C,D,D1,D2:RETURN

1510 CIRCLE(X,Y),H,C:PAINT(X,Y),C,C:CIRCLE(X1,Y1),H1,C1:PAINT(X1,Y1),C1,C1:RETUR
N:'ﾐｶｽﾞｷ
1520 POLY(X,Y),H,C,144,35,900:POLY(X,Y),H/2,C1,144,35,900:PAINT(X,Y),C,C:RETURN:
'ﾋﾀﾞﾘﾎｼ
1530 WIDTH80:COLOR4:CLS4:CSIZE 3:LOCATE 24,0:PRINT #0,"ｾｶｲ ﾉ ｸﾆｸﾞﾆ":PALET:COLOR7
:LOCATE28,3:PRINT"Hudsonsoft";
1540 COLOR5:CSIZE3:LOCATE2,6:PRINT#0,"1  ｱｼﾞｱｼｭｳ":LOCATE40,6:PRINT#0,"2  ﾀｲﾖｳｼｭｳ
":LOCATE2,10:PRINT#0,"3  ｱﾌﾘｶｼｭｳ":LOCATE 40,10:PRINT#0,"4  ﾖｰﾛｯﾊﾟｼｭｳ,ｿﾚﾝ"
1550 LOCATE2,14:PRINT#0,"5  ｷﾀｱﾒﾘｶｼｭｳ":LOCATE40,14:PRINT#0,"6  ﾐﾅﾐｱﾒﾘｶｼｭｳ":A1=6:
GOSUB 1560:RETURN
1560 COLOR6:CSIZE2:LOCATE 14,20:PRINT#0,"ﾄﾞｺﾉ ｸﾆ ｦ ｼﾗﾍﾞﾏｽ ｶ ";:INPUTA
1570 IF A<1ORA>A1 THEN1560 ELSE RETURN
1580 RESTORE1760:CLS4:COLOR7:CSIZE3:LOCATE24,0:PRINT#0,"ｱｼﾞｱ ﾉ ｸﾆｸﾞﾆ｡":PALET
1590 COLOR7:FORI=1TO18:READA$:LOCATE1,2+I:PRINTA$:NEXT:A1=18:GOSUB1560:CLS4:PALE
T:WIDTH40
1600 ON A GOSUB40,50,70,80,90,110,120,210,240,250,270,280,300,310,330,350,380,39
0:GOSUB1830:RETURN
1610 RESTORE1770:CLS4:COLOR7:CSIZE3:LOCATE22,0:PRINT#0,"ﾀｲﾖｳｼｭｳ ﾉ ｸﾆｸﾞﾆ｡":PALET
1620 COLOR7:FORI=1TO4:READA$:LOCATE1,2+I:PRINTA$:NEXT:A1=4:GOSUB1560:CLS4:PALET:
```

```
WIDTH40
1630 ON A GOSUB410,430,450,460:WINDOW(0,0)-(319,199):GOSUB1830:RETURN
1640 RESTORE1780:CLS4:COLOR7:CSIZE3:LOCATE24,0:PRINT#0,"ｱﾌﾘｶ ﾉ ｸﾆｸﾞﾆ｡":PALET
1650 COLOR7:FORI=1TO11:READA$:LOCATE1,2+I:PRINTA$:NEXT:FORI=1TO10:READA$:LOCATE4
1,2+I:PRINTA$:NEXT:A1=21:GOSUB 1560:CLS4:PALET:WIDTH40
1660 ON A GOSUB470,490,500,510,520,550,570,580,590,600,620,630,640,660,670,680,6
90,700,710,720,730:GOSUB1830:RETURN
1670 RESTORE1800:CLS4:COLOR7:CSIZE3:LOCATE24,0:PRINT#0,"ﾖｰﾛｯﾊﾟﾉ ｸﾆｸﾞﾆ｡":PALET
1680 COLOR7:FORI=1TO9:READA$:LOCATE1,2+I:PRINTA$:NEXT:FORI=1TO9:READA$:LOCATE41,
2+I:PRINTA$:NEXT:A1=18:GOSUB1560:CLS4:PALET:WIDTH40
1690 ON A GOSUB740,750,760,870,880,890,900,920,930,940,950,960,970,980,990,1000,
1010,1020:GOSUB1830:RETURN
1700 RESTORE1810:CLS4:COLOR7:CSIZE3:LOCATE24,0:PRINT#0,"ｷﾀｱﾒﾘｶ ﾉ ｸﾆｸﾞﾆ｡":PALET
1710 COLOR7:FORI=1TO7:READA$:LOCATE1,2+I:PRINTA$:NEXT:A1=7:GOSUB1560:CLS4:PALET:
WIDTH40
1720 ON A GOSUB1050,1080,1100,1120,1140,1150,1160:GOSUB1830:RETURN
1730 RESTORE1820:CLS4:COLOR7:CSIZE3:LOCATE24,0:PRINT#0,"ﾐﾅﾐｱﾒﾘｶ ﾉ ｸﾆｸﾞﾆ｡":PALET
1740 COLOR7:FORI=1TO4:READA$:LOCATE1,2+I:PRINTA$:NEXT:A1=4:GOSUB1560:CLS4:PALET:
WIDTH40
1750 ON A GOSUB1170,1190,1210,1220:GOSUB1830:RETURN

1760 DATA"  1: ﾆﾎﾝ","  2: ｲｴﾒﾝ･ｱﾗﾌﾞﾁｮｳﾜｺｸ","  3: ｲｽﾗｴﾙ","  4: ｲﾗｸ","  5: ｲﾝﾄﾞ","
  6: ｲﾝﾄﾞﾈｼｱ","  7: ｶﾝｺｸ","  8: ｶﾝﾎﾞｼﾞｱ","  9: ｼﾘｱ"," 10: ｼﾝｶﾞﾎﾟｰﾙ"," 11: ﾀｲ","
12: ﾁｭｳｺﾞｸ"," 13: ﾄﾙｺ"," 14: ﾊﾟｷｽﾀﾝ"," 15: ﾊﾞｰﾚﾝ"," 16: ﾏﾚｰｼｱ"," 17: ﾓﾙｼﾞﾌﾞ"," 1
8: ﾖﾙﾀﾞﾝ"
1770 DATA"  1: ｵｰｽﾄﾗﾘｱ","  2: ﾂﾊﾞﾙ","  3: ﾄﾝｶﾞ","  4: ﾆｭｰｼﾞｰﾗﾝﾄﾞ"
1780 DATA"  1: ｱﾙｼﾞｪﾘｱ","  2: ｴﾁｵﾋﾟｱ","  3: ｶﾒﾙｰﾝ","  4: ｷﾞﾆｱ","  5: ｹﾆｱ","  6:
ｺﾓﾛ","  7: ｻﾝﾄﾒﾌﾟﾘﾝｼﾍﾟ","  8: ｼｴﾗﾚｵﾈ","  9: ｽｰﾀﾞﾝ"," 10: ｾｲｼｪﾙ"," 11: ｾﾈｶﾞﾙ"," 1
2: ｿﾏﾘｱ"," 13: ﾀﾝｻﾞﾆｱ"," 14: ﾁｭｳｵｳｱﾌﾘｶ"," 15: ﾁｭﾆｼﾞｱ"," 16: ﾄｰｺﾞ"," 17: ﾍﾞﾅﾝ","
18: ﾓｰﾘﾀﾆｱ"
1790 DATA" 19: ﾓﾛｯｺ"," 20: ﾘﾋﾞｱ"," 21: ﾘﾍﾞﾘｱ"
1800 DATA"  1: ｱｲｽﾗﾝﾄﾞ","  2: ｱｲﾙﾗﾝﾄﾞ","  3: ｲｷﾞﾘｽ","  4: ｲﾀﾘｱ","  5: ｵｰｽﾄﾘｱ","
  6: ｵﾗﾝﾀﾞ","  7: ｷﾞﾘｼｱ","  8: ｽｲｽ","  9: ｽｳｪｰﾃﾞﾝ"," 10: ﾁｪｺ"," 11: ﾃﾞﾝﾏｰｸ"," 12:
 ﾆｼﾄﾞｲﾂ"," 13: ﾉﾙｳｪｰ"," 14: ﾌｨﾝﾗﾝﾄﾞ"," 15: ﾌﾗﾝｽ"," 16: ﾍﾞﾙｷﾞｰ"," 17: ﾕｰｺﾞｽﾗﾋﾞｱ",
" 18: ｿﾚﾝ"
1810 DATA"  1: ｱﾒﾘｶ","  2: ｶﾅﾀﾞ","  3: ｼﾞｬﾏｲｶ","  4: ｾﾝﾄﾙｼｱ","  5: ﾊﾟﾅﾏ","  6: ﾊ
ﾞﾊﾏ","  7: ﾎﾝｼﾞｭﾗｽ"
1820 DATA"  1: ｶﾞｲｱﾅ","  2: ｽﾘﾅﾑ","  3: ﾁﾘ","  4: ﾌﾞﾗｼﾞﾙ"
1830 COLOR6:LINE(0,110)-(319,180),PSET,B:LINE(0,138)-(319,138):COLOR7:CSIZE2:LOC
ATE2,14:PRINT#0,Q$:LOCATE2,16:PRINTQ1$:LOCATE2,18:PRINT"ｼｭﾄ  : ";Q2$:LOCATE2,19:
IF Q=0 THEN PRINT"ﾒﾝｾｷ : ";Q3$;",000 Km2" ELSE PRINT"ﾒﾝｾｷ : ";Q3$;" Km2"
1840 LOCATE2,20:IF Q1=0 THEN PRINT"ｼﾞﾝｺｳ: ";Q4$;",000 ﾆﾝ" ELSE PRINT"ｼﾞﾝｺｳ: ";Q4
$
1850 LOCATE2,21:IF Q2=0 THEN PRINT"GNP  : $";Q5$;",000,000" ELSE PRINT "GNP  : "
;Q5$
1860 Q=0:Q1=0:Q2=0:LINE(160,1)-(318,99),PSET,6,B:CSIZE 3:LINE(0,4)-(39,9),"",BF
1870 LOCATE0,24:PRINT"Push any key";:WHILE INKEY$="":LOCATE 22,3:PRINT"ﾆﾎﾝ ｼﾞｶﾝ"
:LOCATE 20,8:TH=VAL(LEFT$(TIME$,2))+TT:IFTH>24THENTH=TH-24ELSEIFTH<0THENTH=TH+24
1880 LOCATE22,7:PRINTQ2$;" ｼﾞｶﾝ";:LOCATE22,8:PRINT#0,RIGHT$("0"+MID$(STR$(TH),2)
,2);MID$(TIME$,3):LOCATE22,4:PRINT#0,TIME$:WEND:RETURN
```

資料
外務省情報文化局(財)世界の動き社刊　世界の国一覧表（1982年版）

（写真１）

（写真２）

（写真３）

これはグラフィック命令を使って，何か楽しいプログラムを作ってみようという事で，先生が旗のプログラムを考えてくれたものです。世界72カ国の旗を入れて，しかも時差を計算して時計も入れちゃったんですよ。すごいでしょう。

その他，人口とか，面積とか，GNP も出てきますよ。

さて，『世界の国々』の調べ方です。

まず，このような表示が現れますから，調べたい地域の番号を Push！（写真１）

じゃあ，今はとりあえずアジアを見てみますヨ。

次には，このようにアジア各国のメニューが出て来ます。どれにしようかナ。（写真２）

さあ，どうですか。苦労に苦労を重ね，何度もやり直して作りあげた旗のグラフィックなのです。きれいでしょ。もちろん時計や資料も正確ですからネ……。

ところで，次にプログラムの流れを見ていきましょう。延々とプログラムが続いていて，すごく難しそうだけど，どうなっているんでしょうね。

実は，このプログラムのメインルーチンは40番だけなのです。この一行で流れを決めてしまっているのだから，スゴイでしょ。

30番でイニシャライズ（初期化）が行われ，40番でサブルーチンへ飛ばし，GOTO 40 で何度もくり返すわけです。

つまり，これだけ長いプログラムも，ほとんどがデータの部分であって，プログラム自体の流れは，けっこう単純だというわけ。あまり恐怖心を持たなくても大丈夫ですよ。

さて，50番を見てください。この行には日本のデータが入れてあります。（写真３）

まずＣ＝7 で色を白と指定し，1280番のサブルーチンへ飛ばして，白い旗を描かせます。次に中心座標と色（ここでは赤）を指定してやり，今度は，円を描いてぬりつぶすサブルーチン（1310，1320）へそれぞれ飛ばすわけです。もちろん，サブルーチンの後ろには RETURN がついていますから，すぐに50番へ戻ってきますよね。

それから，日本について順番に入っているデータを読んで表示してくれるわけです。データは，国名（カタカナ），国名（英語），首都，面積，人口，GNP，日本時間とそれに各国の首都時間です。

1カ国分の流れは大体こんな感じなんです。描くのが難しい国旗もありましたが，サブルーチン処理をできるだけ利用してクリアしました。

たとえば，大洋州の国旗には，すべて英国旗が縮小されて描かれているので，WINDOW 命令を使って，英国旗を$\frac{1}{4}$にしたサブルーチンを作りました。その他では，星の形を書くサブルーチン，縦三つに分かれている旗（フランスやイタリア等），横三つに分かれている旗などという形で，まとめられるものは全てサブルーチンを作ってまとめてあります。

50番から1250番まで，このような流れで72カ国分のデータが入っています。

1260番から1520番までは，グラフィック命令が入れてあり，旗を描くのに必要な命令が GOSUB-RETURN でどんどん使われます。

1530番から1750番までは，最初にどの国を調べようか決めるまでの画面に現れる内容です。地域別にわけた国のリストが，出てくる部分のプログラムですよね。

1760番から1820番は，国名のデータで，地域別に分けて入れられています。

1830番から1860番あたりは，人口や面積に格差が生じた国の数字の表示の仕方を，0を減らしたりして調節している部分です。

そして最後の2行（1870, 1880）で，日本時間と，各国の首都時間を，時差を計算して入れています。

一通りの流れは，こんな感じなんですが，どうでしょうねぇ。プログラムの中にも，そこで何をやっているのかわかるように，1ボックスフルなんていう風に入れてありますから，見てもらえれば分かるんじゃないかなあ。

```
** Personal Telephone List **

  * ﾒｲﾚｲ ｲﾁﾗﾝﾋﾟｮｳ *

 KEY1:File ｹﾝｻｸ
 KEY2:File ﾘｽﾄ
 KEY3:File ﾆｭｳﾘｮｸ            ｻｲﾀﾞｲ ﾗｲﾝ: 256
 KEY4:File ﾍﾝｺｳ              ｼｮｳ  ﾗｲﾝ : 0
 KEY5:File ﾂｲｶ               ﾌﾟﾘﾝﾀｰ   : OFF
 KEY6:File LOAD
 KEY7:File SAVE              END :ESC KEY
 KEY8:ﾌﾟﾘﾝﾀｰON/OFF

INPUT   [KEY1]-[KEY8]
```

ﾅﾏｴ	TEL.	ﾒ ﾓ
HUDSON SOFT	011-821-1538	ｻｯﾎﾟﾛ ﾎﾝｼｬ
HUDSON SOFT	03-234-4996	ﾄｳｷｮｳ ｴｲｷﾞｮｳｼｮ
HUDSON SOFT	06-251-1945	ｵｵｻｶ ｴｲｷﾞｮｳｼｮ
HUDSON USA	415-845-1416	BERKELEY CA94704
CQ HUDSON	011-821-1189	ｱﾏﾁｭｱﾑｾﾝ ﾉ ｾﾝﾓﾝﾃﾝ
ｼﾅｶﾞﾜ ﾕﾘ	03-234-4994	ｹｲｵｳﾀﾞｲｶﾞｸ

```
10 REM
20 REM
30 REM      TELEPHONE LIST PROGRAM
40 REM
50 REM     Copyright (C) 1982,10,20
60 REM
70 REM      By Hudsonsoft  Y.Kudo
80 REM
90 REM
100 WIDTH 40:CONSOLE 0,25:COLOR 7,0:SCREEN 0,0,1:WINDOW(0,0)-(319,199):PALET:CLS
0
110 Z$=CHR$(26):E$=CHR$(5):FORI=1TO8:KEYI,RIGHT$(STR$(I),1):NEXT:KEY(9)ON:KEY(10
)ON:GOTO930
120 CLS:COLOR2:PRINT"  ** Personal Telephone List **":COLOR5:LOCATE4,2:PRINT"* ﾒ
ｲﾚｲ ｲﾁﾗﾝﾋﾞｮｳ *"
130 COLOR4:LOCATE4,4:PRINT"KEY1";:COLOR7:PRINT":File ｹﾝｻｸ"
140 COLOR4:LOCATE4,6:PRINT"KEY2";:COLOR7:PRINT":File ﾘｽﾄ"
150 COLOR4:LOCATE4,8:PRINT"KEY3";:COLOR7:PRINT":File ﾆｭｳﾘｮｸ"
160 COLOR4:LOCATE4,10:PRINT"KEY4";:COLOR7:PRINT":File ﾍﾝｺｳ"
170 COLOR4:LOCATE4,12:PRINT"KEY5";:COLOR7:PRINT":File ｻｲｶ"
180 COLOR4:LOCATE4,14:PRINT"KEY6";:COLOR7:PRINT":File LOAD"
190 COLOR4:LOCATE4,16:PRINT"KEY7";:COLOR7:PRINT":File SAVE":PRINTTAB(25);"END :E
SC KEY"
200 COLOR4:LOCATE4,18:PRINT"KEY8";:COLOR7:PRINT":ﾌﾟﾘﾝﾀｰON/OFF"
210 COLOR5:LOCATE24,10:PRINT"ｻｲﾀﾞｲ ﾗｲﾝ:";MXL:LOCATE24,12:PRINT"ｼｮｳ  ﾗｲﾝ :";USL:L
OCATE24,14:PRINT"ﾌﾟﾘﾝﾀｰ   : ";:IFPR=0THENPRINT"OFF"ELSE PRINT"ON "
220 LINE(24,28)-(168,156),PSET,6,B:LINE(184,68)-(312,128),PSET,6,B:LINE(184,132)
-(312,148),PSET,6,B
230 IF USL=0 THEN LOCATE 5,22:COLOR6:PRINT"ﾃﾞｰﾀｰ ｶﾞｱﾘﾏｾﾝ  ";:COLOR2:PRINT"LOAD o
r INPUT"
240 COLOR5:LOCATE5,20:PRINT"INPUT ";:COLOR5:PRINT"[";:COLOR4:PRINT"KEY1";:COLOR5
:PRINT"]-[";:COLOR4:PRINT"KEY8";:COLOR5:PRINT"]":FORI=0TO30:A$=INKEY$:IFA$=""THE
NNEXT:GOTO260
250 GOTO 280
260 IF PW=0 THEN LOCATE5,22:PRINT"                    "
270 LOCATE5,20:PRINT"     ":FORI=0TO30:A$=INKEY$:IFA$=""THENNEXT:GOTO230
280 IFA$=CHR$(27)THEN1090 ELSE A=VAL(A$):IFA=0THEN230 ELSE ON A GOTO 970,450,300
,550,740,770,840,910
290 GOTO 210

300 SCREEN0,0,0:COLOR7:CLS:CLS0:PRINT"** FILE ﾆｭｳﾘｮｸ ﾓｰﾄﾞ**"
310 LOCATE0,4:PRINT"Ready ? [Y or N]_";CHR$(29);" ":A$=INKEY$:IFA$=""THEN310
320 IFA$="N" THEN 120 ELSE IF A$="Y" THEN 330 ELSE GOTO310
330 USL=0
340 COLOR7:CLS:CLS0:PRINT"** FILE ﾆｭｳﾘｮｸ ﾓｰﾄﾞ**":USL=USL+1:LOCATE0,2:PRINT"DATA
No.";USL:FOR I=1 TO MXF
350 LOCATE0,I*2+2:PRINT"No.";I;N$(I);":";:IFI=3 THENPRINTSPC(60);:LOCATE11,8
360 INPUT F$(I,USL):IF I=1 OR I=2 THEN IF LEN(F$(I,USL))>14 THEN LOCATE 0,I*2+2:
PRINTSPC(39):GOTO350
370 IF I=3 THEN IF LEN(F$(I,USL))>50 THEN LOCATE0,8:PRINTSPC(39):PRINTSPC(40):GO
TO350
```

```
380 NEXT
390 LOCATE0,23:PRINTSPC(39);:LOCATE0,22:PRINT"KEY1:ﾂｷﾞﾍ KEY2:ﾃｲｾｲ KEY3:ﾆｭｳﾘｮｸｵﾜﾘ
 ▒";CHR$(29);" ";:A$=INKEY$:IFA$=""THEN390 ELSE IF A$="1"THEN340 ELSE IF A$="2"
THEN 400 ELSE IF A$="3" THEN 120 ELSE 390
400 CONSOLE23,2:LOCATE0,23:PRINTSPC(39):LOCATE0,23:PRINT"ﾃｲｾｲ   No.(0=Can):";:IN
PUT CN$:IFCN$=""THEN400 ELSE CN=VAL(CN$):IFCN=0THENCONSOLE0,25:GOTO390 ELSE IF C
N>MXF THEN 400
410 LOCATE0,23:PRINTSPC(39):LOCATE0,23:PRINT N$(CN);:INPUT":";F$(CN,USL)
420 IFCN=1 OR CN=2 THEN IF LEN(F$(CN,USL))>14 THEN 410
430 IF CN=3 THEN IF LEN(F$(CN,USL))>50 THEN 410
440 CONSOLE0,25:LOCATE0,3:PRINTCHR$(26):FORI=1TOMXF:LOCATE0,I*2+2:PRINT"No.";I;N
$(I);":";F$(I,USL):NEXT:GOTO390
450 SCREEN0,0,0:WIDTH80:COLOR2:PRINT"** FILE ﾘｽﾄ ﾓｰﾄﾞ **":COLOR7:IF PR=1 THEN 51
0
460 FOR I=1TOMXF:PRINT TAB(T(I));N$(I);:NEXT:PRINT
470 FOR I=1TOUSL:FORJ=1TOMXF:PRINTTAB(T(J));F$(J,I);:NEXT:PRINT
480 P=I/20-INT(I/20):IFP=0 THEN LOCATE0,23:PRINT"KEY1:Menu  KEY2:ﾂﾂﾞｹﾙ ▒";CHR$(2
9);" ";:A$=INKEY$:IFA$=""THEN 480 ELSE IFA$="1"THEN120 ELSE IFA$="2"THEN CLS ELS
EGOTO 480
490 NEXT:PRINT"FILE ｼｭｳﾘｮｳ "
500 LOCATE0,23:PRINT"KEY1:Menu KEY2:ﾓｳｲﾁﾄﾞ ▒";CHR$(29);" ";:A$=INKEY$:IFA$=""THE
N500ELSEIFA$="1"THENWIDTH40:GOTO120ELSEIFA$="2"THEN450ELSEGOTO500
510 OPEN"O",#1,"LPT0:":FORI=1TOMXF:PRINT#1,TAB(20*(I-1));N$(I);:NEXT:PRINT#1,""
520 FOR I=1TOUSL:FORJ=1TOMXF:PRINT#1,TAB(20*(J-1));F$(J,I);:NEXT:PRINT#1
530 NEXT:CLOSE
540 LOCATE0,23:PRINT"KEY1:Menu  KEY2:ｹｲｿﾞｸ ▒";CHR$(29);" ";:A$=INKEY$:IFA$=""THE
N540ELSEIFA$="1"THENWIDTH40:GOTO120ELSEIFA$="2"THEN510ELSEGOTO540
550 SCREEN0,0,0:CLS:CLS0:LOCATE0,0:COLOR6:PRINT"** ﾍﾝｺｳ ﾓｰﾄﾞ  **":COLOR 7
560 FOR I=1TOMXF:LOCATE0,I*2+2:PRINT"KEY";I;N$(I):NEXT

570 LOCATE0,23:PRINT"ｹﾝｻｸ ｱﾝﾅｲ   (0=Menu)▒";CHR$(29);" ";:SN$=INKEY$:IFSN$=""THE
N570 ELSE SN=VAL(SN$):IFSN$="0"THENCONSOLE0,25:GOTO120 ELSEIFSN=0THEN570ELSEIFSN
>MXF THEN570
580 LOCATE0,23:PRINTZ$:LOCATE0,SN*2+2:COLOR2:PRINT"KEY";SN;N$(SN):COLOR7:LOCATE0
,23:PRINTN$(SN);:INPUT":";SD$
590 FOR I=1TOUSL:J=INSTR(F$(SN,I),SD$):IF J<>0 THENK=I:GOTO610
600 NEXT:LOCATE0,18:PRINT"ﾌｧｲﾙ ｶﾞｱﾘﾏｾﾝ":LOCATE0,23:PRINTZ$:FORI=0TO2000:NEXT:LOC
ATE0,18:PRINTE$:GOTO550
610 CONSOLE0,24:CLS:CLS0:LOCATE0,0:COLOR6:PRINT"  ** ﾍﾝｺｳ ﾓｰﾄﾞ  **":COLOR7
620 FORJ=1TOMXF:LOCATE0,J*2+2:PRINT"KEY";J;" ";F$(J,K):NEXT
630 LOCATE0,23:PRINTCHR$(26);
640 LOCATE0,23:PRINT"KEY1:ﾂｷﾞﾉ ﾃﾞｰﾀ  KEY2:ﾍﾝｺｳ  KEY3:Menu ▒";CHR$(29);" ";:A$=IN
KEY$:IFA$=""THEN640ELSE IFA$="1"THEN650ELSEIFA$="2"THEN670ELSEIFA$="3"THENCONSOL
E0,24:GOTO120ELSEGOTO640
650 FORI=K+1TOUSL:J=INSTR(F$(SN,I),SD$):IFJ<>0THEN K=I:GOTO610
660 NEXT:LOCATE2,18:PRINT"ｹﾝｻｸ ｵﾜﾘ  ":FORI=1TO1000:NEXT:LOCATE0,18:PRINTSPC(39):
GOTO630
670 LOCATE0,23:PRINTCHR$(26)
680 LOCATE0,23:PRINT"ﾍﾝｺｳ ｶｼｮ (0=Can) ▒";CHR$(29);" ";:C$=INKEY$:IFC$=""THEN680E
LSEC=VAL(C$)
690 IFC$="0"THEN630ELSEIFVAL(C$)=0ORC>4ORC<0 THEN 680
```

```
700 CONSOLE23,2:LOCATE0,23:PRINTCHR$(26):LOCATE0,23:PRINTN$(C);:INPUT":";X$:F$(C
,K)=X$
710 IFC=1 OR C=2 THEN IF LEN(X$)>14 THEN 700
720 IFC=3 THEN IF LEN(X$)>50 THEN 700
730 LOCATE0,23:PRINTCHR$(26);:CONSOLE0,25:GOTO610
740 SCREEN0,0:COLOR7:CLS:CLS0:PRINT"** File ﾂｲｶ ﾓｰﾄﾞ **"
750 LOCATE0,4:PRINT"Ready ? [Y or N] ▒";CHR$(29);" ":A$=INKEY$:IFA$=""THEN750
760 IFA$="N" THEN 120 ELSE IF A$="Y" THEN 340 ELSE GOTO 750
770 SCREEN0,0,0:CLS:CLS0:PRINT"** FILE LOAD ﾓｰﾄﾞ **"
780 COLOR7:LOCATE2,4:PRINT"ﾏｽﾀｰﾃｰﾌﾟ ｦ ﾚｺｰﾀﾞｰ ﾆ ｾｯﾄ ｼ､"
790 LOCATE2,6:PRINT"RETURNｷｰ ｦ ｵｼﾃｸﾀﾞｻｲ｡"
800 LOCATE2,8:PRINT"ﾀﾀﾞｼ､ESC ｷｰﾃﾞｺﾉ ﾓｰﾄﾞﾊ ｷｬﾝｾﾙｻﾚﾏｽ｡ ▒";CHR$(29);" ":A$=INKEY$:I
FA$=""THEN800 ELSE IF A$=CHR$(13) THEN 810 ELSE IF A$=CHR$(27)THEN120ELSEGOTO800

810 PRINT:PRINT"  ﾃﾞｰﾀ ﾖﾐｺﾐﾁｭｳ ...":OPEN"I",#1,"DATA-TEL"
820 INPUT#1,USL
830 FORI=1TOUSL:FORJ=1TO4:INPUT#1,F$(J,I):NEXT:NEXT:CLOSE:GOTO120
840 SCREEN0,0,0:CLS:CLS0:PRINT"** FILE SAVE ﾓｰﾄﾞ **"
850 COLOR7,0:LOCATE2,4:PRINT"ｱﾀﾗｼｲ ﾃｰﾌﾟｦ ﾚｺｰﾀﾞｰ ﾆ ｾｯﾄ ｼ､"
860 LOCATE2,6:PRINT"RETURNｷｰ ｦ ｵｼﾃｸﾀﾞｻｲ "

870 LOCATE2,8:PRINT"ﾀﾀﾞｼ､ESC ｷｰ ﾃﾞｺﾉ ﾓｰﾄﾞﾊ ｷｬﾝｾﾙｻﾚﾏｽ｡ ▒";CHR$(29);" ":A$=INKEY$:
IFA$=""THEN870ELSEIFA$=CHR$(13)THENPRINT:PRINT"  ﾃﾞｰﾀｰ ｶｷｺﾐﾁｭｳ ...":GOTO880ELSEI
FA$=CHR$(27)THEN120ELSEGOTO870
880 OPEN"O",#1,"DATA-TEL"
890 PRINT#1,USL
900 FOR I=1TOUSL:FORJ=1TO4:PRINT#1,F$(J,I):NEXT:NEXT:CLOSE:GOTO120
910 IF PR=1THENPR=0ELSEPR=1
920 GOTO210
930 MXL=256
940 DIMF$(4,MXL),N$(3),T(3):MXF=3
950 FORI=1TO3:READN$(I),T(I):NEXT:GOTO120
960 DATA"ﾅﾏｴ ",0,TEL.,15,"ﾒ ﾓ ",29
970 SCREEN0,0,0:CLS:CLS0:COLOR6:PRINT"** ｹﾝｻｸ ﾓｰﾄﾞ **":COLOR7
980 FORI=1TOMXF:LOCATE0,I*2+2:PRINT"KEY";I;N$(I):NEXT
990 LOCATE0,23:PRINT"ｹﾝｻｸ ﾅｲﾖｳ (0=Menu) ▒";CHR$(29);" ";:SN$=INKEY$:IF SN$="" TH
EN990 ELSE IFSN$="0" THEN CONSOLE0,24:GOTO120 ELSE SN=VAL(SN$):IF(SN=0)OR(SN>4)T
HEN 990
1000 LOCATE0,23:PRINTZ$;
1010 LOCATE0,SN*2+2:COLOR2:PRINT"KEY";SN;N$(SN):COLOR7:LOCATE0,23:PRINTN$(SN);:I
NPUT":";SD$:CONSOLE0,24
1020 FORI=1TOUSL:J=INSTR(F$(SN,I),SD$):IFJ<>0 THENK=I:GOTO1040
1030 NEXT:LOCATE5,18:COLOR2:PRINT"ﾐﾂｶﾘﾏｾﾝ  ":LOCATE0,23:PRINTZ$:COLOR7:FORI=0TO2
000:NEXT:GOTO970
1040 CONSOLE0,24:CLS:CLS0:COLOR6:PRINT"  ** ｹﾝｻｸ ﾓｰﾄﾞ **":COLOR7
1050 FORJ=1TOMXF:LOCATE0,J*2+2:PRINTN$(J)": ";F$(J,K):NEXT
1060 LOCATE0,23:PRINT"KEY1:ﾂｷﾞﾍ  KEY2:ｹﾝｻｸ ﾓｰﾄﾞ  KEY3:Menu ▒";CHR$(29);" ";:A$=I
NKEY$:IFA$=""THEN1060 ELSEIFA$="1"THEN 1070 ELSEIFA$="3"THEN CONSOLE0,25:GOTO120
 ELSEIFA$="2"THEN970 ELSEGOTO1060
1070 FORI=K+1TOUSL:J=INSTR(F$(SN,I),SD$):IFJ<>0THENK=I:GOTO1040
```

```
1080 NEXT:LOCATE2,18:PRINT"ｹﾝｻｸ ｵﾜﾘ":FORI=1TO1000:NEXT:LOCATE0,18:PRINTSPC(39):G
OTO1060
1090 CLS:CLS0:COLOR7,0:CONSOLE0,24:KEY1,"AUTO"+CHR$(13):KEY2,"CONSOLE ":KEY3,"KE
Y":KEY4,"LIST"+CHR$(13):KEY5,"RUN"+CHR$(13):KEY6,"LOAD"+CHR$(13):KEY7,"WIDTH ":K
EY8,"CHR$(":KEY(9)OFF:KEY(10)OFF:END
```

これはテレフォン・リストのプログラムです。コンピューターに，電話帳の役目を果たしてもらいましょう。

まず，ここでもプログラムの流れを見ていきましょうね。

100番から110番までは，初期化を行っている部分です。100番に **CONSOLE** という見慣れぬ命令が入ってますが，これは，画面を縦の範囲でのみ指定するものです。ここではCONSOLE 0，24ですから，0行目から24行目まで，つまり縦一杯に画面をとったわけです。

120番から290番までがメインルーチンで，130番から190番までは，どのKeyが何の働きをするかの部分で，検索，リスト，入力などを使いやすく分類してあります。

280番を見てください。CHR$(27)とはエスケープ ESC の事で，もしA$が ESC ならば，1090番へ飛びます。

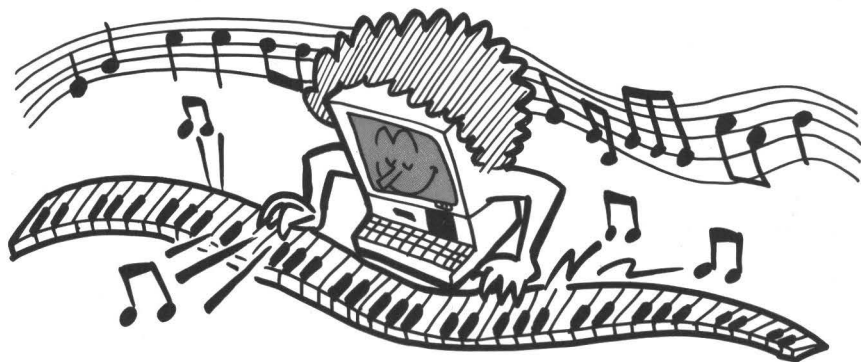
1090番は，プログラムのEND部分になりますから， ESC を押せば，新たにリストを見れる状態に戻るわけですね。

メインルーチンの残りの部分は，それぞれのFileを調べる時に必要な，質問や説明書きを表示する等のプログラムです。

300番から1080番までは，大きく分けて7つの部分に別れます。特にサブルーチン処理をしてあるわけではありませんが，KEY 1～KEY 7までのFile別にまとめてあるのです。

例えば，970番から1080番までは，検索モードで名前，Tel，メモのどれからでも検索を行ってくれる部分です。

その他では，300番から440番が入力モード，450番から540番までがリストモード，550番から730番までが変更モード，740番から760番までが追加モード，というようになっているのが分かるでしょ。

930番から960番までは，ディメンジョンの設定と，ナマエ，TELなどの表題を与えている部分です。そして1090番がEND。

```
10 REM
20 REM
30 REM    Music sample  Hudsonsoft
40 REM
50 REM    (C) 1982,10,9  T.Takahashi
60 REM
70 REM
80 CFLASH0:CLEAR:CLS:WIDTH 40:COLOR 7:LOCATE12,2:PRINT"MUSIC SAMPLE"
90 COLOR 4:LOCATE 6,8:PRINT"SWAN (ﾊｸﾁｮｳ ﾉ ﾐｽﾞｳﾐ)･････1"
100 COLOR 5:LOCATE 6,10:PRINT"Menuet (ｱﾙﾙ ﾉ ｵﾝﾅ ﾖﾘ)････2"
110 COLOR 6:LOCATE 6,12:PRINT"MARCH (ｵﾓﾁｬ ﾉ ﾏｰﾁ)･･･････3"
120 COLOR 3:LOCATE 6,14:PRINT"MARCH-2 (ﾘｽﾞﾑ ﾂｷ)････････4"
130 COLOR 2:LOCATE11,20:PRINT"Push Any key"
140 A$=INKEY$
150 IF A$=""THEN A$=" "
160 A=VAL(A$):IF A>0 AND A<5 THEN GOSUB 1160 ELSE 140
170 GOTO 80
180 REM
190 REM          Swan
200 REM
210  SOUND7,&B111000
220 TEMPO 100
230 PLAY"V15:V12:V13"
240 PLAY"O3-B2#FBO4D2#FB+DB#FD-B-#F"
250 PLAY"O5#F7-B3#CDE:O4#F7G:O4D7E"
260 PLAY"O5#F6D3#F6D3:O4#F9:O4D9"
270 PLAY"O5#F6-B3O4+D3BG+D:O4#F7F5E:O4D7-B5-B3-#A3"
280 PLAY"O4B7B3+E+D+#C:O4D5G#FE:O4-B5BAG"
290 PLAY"O5#F7-B3#CDE:O4D7D5-B:O4#F7F5E"
300 PLAY"O5#F6D3#F6D3:O4-B7#C5D:O3+D5#G#AB"
310 PLAY"O5#F6-B3O4+D3BG+D:O4E5DDE:O4#A5BBB3#A"
320 PLAY"O4B7R5B5:O4D5#FB#F:O4-B5D#FD"
330 PLAY"O5#C5DE#F3G:O4A9:O4E9"
340 PLAY"O5A6G3#F5G3A:O4A9:O4#F9"
350 PLAY"O5B6A3G5A3B:O4B9:O4G9"
360 PLAY"O5+#C6B3#FD#C-B:O4+#C7#A5#F:O4#G7#F5D"
370 PLAY"O5#C5DE#F3G:O4A9:O4E9"
380 PLAY"O5A6G3#F5G3A:O4A9:O4#F9"
390 PLAY"O5B6A3G5A3B:O4B9:O4G9"
```

```
400 PLAY"O5+C6G3E5G3+C:O4+C9:O4G9"
410 PLAY"O5+#C6#G3+#C6#F3:O4+#C9:O4#G7#A"
420 PLAY"O5B7R8:O4+D7:O4B"
430 RETURN
440 REM
450 REM          Menuet
460 REM
470  SOUND15,&H38
480 TEMPO 100
490 PLAY"V10:V15"
500 PLAY"O4-#D3-#A#D-#AG-#A"
510 PLAY"O4-#D3-#A#D-#AG-#A"
520 PLAY"O4-#D3-#A#D-#AF-#A:O5G4R1G4F1#DFG#G"
530 PLAY"O4-#D3-#AG-#AG-#A:O5#A3GO6#D-#AGR"
540 PLAY"O4-#A3#AD-#AD-#A:O5F4R1F4G1F#DDC"
550 PLAY"O4-#A3#AF#AD#A:O5-#A3DFD#AR"
560 PLAY"O4-C3GCG#DG:O5#D4R1#D5F1#DD#D"
570 PLAY"04-#G3FCF#DF:O5F3G#G#A+CR"
580 PLAY"O4-#A#GD#A#G#A:O5F4R1F5G0FE1FG"
590 PLAY"O4-#D3-#A#D-#AG-#A:O5#D6R6"
600 PLAY"O4-#D3-#A#D-#AF-#A:O5G4R1G4F1#DFG#G"
610 PLAY"O4-#D3-#A#D-#A#D-#A:O5#A1G#AGO6#D-#A#D-#AG3R3"
620 PLAY"O3#A3O4#A3D-#A#D-#A:O5F4R1F4G1F#DDC"
630 PLAY"O3#A3O4#A3F#AD#A:O5-#A1-F-#ADFDFD#A3R3"
640 PLAY"O3C3G+CG+#DG:O5#D4R1#D5F1#DD#D"
650 PLAY"O3#G3D+CF+#DF:O5F1G#G#A+C4R1+C1#A#GG"
660 PLAY"O3#A3O4#GD#A+#G#A:O5F2R0F2R0F2R0F2R0GFE1FG"
670 PLAY"O4-#D3-#A#D-#AG-#A:O5#D7R5"
680 PLAY"O4#D5"
690 RETURN
700 REM
710 REM          March
720 REM
730  SOUND7,&B111000
740 TEMPO 130
750 PLAY"V15:V15"
760 PLAY"O5C2R0C2R0C2R0E2R0G2R0G2R0G5:O4+C3BABO5C2R0C2R0C5"
770 PLAY"O5D2R0D2R0D2R0F2R0G2R0G2R0G6R1:O4B3AGBA2R0A2R0G6R1"
780 PLAY"O5G2R0G2R0#G2R0A2R0A2R0A2R0#G2R0:O4G2R0G2R0G2R0G2R0G2R0G2R0G2R0"
790 PLAY"O5G5B+C7:O4G5-BC7"
800 PLAY"O5R3G2R0G2R0F2R0E2R0E2R0E2R0F2R0:O4C3EG+CCEG+C"
810 PLAY"O5G2R0G2R0G2R0G2R0C6R3:O4C3EG+CCEG+C"
820 PLAY"O5R3F2R0F2R0E2R0D2R0D2R0D2R0D2R0:O4-B3DFB-BDFB"
830 PLAY"O5D2R0A2R0#G2R0#G2R0G6R3:O4-B3DFBO5DFAB"
840 PLAY"O5A2R0A2R0A2R0B2R0O6C2R0C4R1C2R0:O4B3AGBAGFE"
850 PLAY"O5B2R0B2R0B4R1A6R2:O4-B3DFABGFD"
860 PLAY"O5G2R0G2R0#G2R0A2R0A2R0A2R0#G2R0:O4G2R0G2R0G2R0G2R0G2R0G2R0G2R0"
870 PLAY"O5G4R1B4R1+C6R3:O4G4R1-B4R1C6R3"
880 PLAY"O5C2R0C2R0C2R0E2R0G2R0G2R0G5:O4+C3BABO5C2R0C2R0C5"
890 PLAY"O5D2R0D2R0D2R0F2R0G2R0G2R0G6R1:O4B3AGBA2R0A2R0G6R1"
```

```
900 PLAY"O5G2R0G2R0#G2R0A2R0A2R0A2R0#G2R0:O4G2R0G2R0G2R0G2R0G2R0G2R0G2R0"
910 PLAY"O5G5B+C7:O4G5B+C7"
920 RETURN


930 REM
940 REM      March+Rizum
950 REM
960  SOUND7,&B11100
970 TEMPO 130
980 PLAY"V15:V15:V15"
990 PLAY"O5C2R0C2R0C2R0E2R0G2R0G2R0G5:O4+C3BABO5C2R0C2R0C5:O7B0R2B0RBRBRBRBR2B0R
BRBRBRBR"
1000 PLAY"O5D2R0D2R0D2R0F2R0G2R0G2R0G6R1:O4B3AGBA2R0A2R0G6R1:O7B0R2B0RBRBRBRBR2B
0RBRBRBRBRBRBR"
1010 PLAY"O5G2R0G2R0#G2R0A2R0A2R0A2R0#G2R0:O4G2R0G2R0G2R0G2R0G2R0G2R0G2R0:O7B0R2
B0RBRBRBRBRBRBRBRBRBRBRBR"
1020 PLAY"O5G5B+C7:O4G5-BC7:O7B1R4B1R4B2"
1030 PLAY"O5R3G2R0G2R0F2R0E2R0E2R0E2R0F2R0:O4C3EG+CCEG+C:O7B0R2B0RBRBR2B0RBRBR2B
0RBRBRBRBRBR"
1040 PLAY"O5G2R0G2R0G2R0G2R0C6R3:O4CEG+CCEG+C:O7B0R2B0RBRBR2B0RBRBR2B0RBRBR2B0RB
R"
1050 PLAY"O5R3F2R0F2R0E2R0D2R0D2R0D2R0D2R0:O4-B3DFB-BDFB:O7B0R2B0RBRBR2B0RBRBR2B
0RBRBRBRBR"
1060 PLAY"O5D2R0A2R0#G2R0#G2R0G6R3:O4-BDFBO5DFAB:O7B0R2B0RBRBR2B0RBRBR2B0RBRBRBR
BR"
1070 PLAY"O5A2R0A2R0A2R0B2R0O6C2R0C4R1C2R0:O4B3AGBAGFE:O7B0R2B0RBRBR2B0RBRBR2B0R
BRBRBRBRBR"
1080 PLAY"O5B2R0B2R0B4R1A6R2:O4-B3DFABGFD:O7B0R2B0RBRBR2B0RBRBR2B0RBRBR2B0RBR"
1090 PLAY"O5G2R0G2R0#G2R0A2R0A2R0A2R0#G2R0:O4G2R0G2R0G2R0G2R0G2R0G2R0G2R0:O7B0R2
B0RBRBR2B0RBRBR2B0RBRBRBR"
1100 PLAY"O5G4R1B4R1+C6R3:O4G4R1-B4R1C6R3:O7B1R4B1R4B"
1110 PLAY"O5C2R0C2R0C2R0E2R0G2R0G2R0G5:O4+C3BABO5C2R0C2R0C5:O7B0R2B0RBRBRBRBR2B0
RBRBRBRBR"
1120 PLAY"O5D2R0D2R0D2R0F2R0G2R0G2R0G6R1:O4B3AGBA2R0A2R0G6R1:O7B0R2B0RBRBRBRBR2B
0RBRBRBR"
1130 PLAY"O5G2R0G2R0#G2R0A2R0A2R0A2R0#G2R0:O4G2R0G2R0G2R0G2R0G2R0G2R0G2R0:O7B0R2
B0RBRBRBRBRBRBRBRBRBRBRBR"
1140 PLAY"O5G5B+C7:O4G5B+C7:O7B1R4B1R4B"
1150 RETURN
1160 ON A GOTO 1180,1260,1340
1170 GOTO 1420
1180 CLS
1190 CFLASH 1:COLOR 4
1200 LOCATE 8,8:PRINT"N O W  P L A Y I N G"
1210 LOCATE11,10:PRINT" SWAN "
1220 LOCATE12,12:PRINT"ﾊｸﾁｮｳ ﾉ ﾐｽﾞｳﾐ"
1230 GOSUB 180
1240 CFLASH 0
1250 RETURN
1260 CLS
```

```
1270 CFLASH 1:COLOR 5
1280 LOCATE 8,8:PRINT"N O W  P L A Y I N G"
1290 LOCATE11,10:PRINT"Menuet"
1300 LOCATE12,12:PRINT"ｱﾙﾙ ﾉ ｵﾝﾅ  ﾖﾘ"
1310 GOSUB 440
1320 CFLASH 0
1330 RETURN
1340 CLS
1350 CFLASH 1:COLOR 6
1360 LOCATE 8,8:PRINT"N O W  P L A Y I N G"
1370 LOCATE11,10:PRINT"MARCH"
1380 LOCATE12,12:PRINT"ｵﾓﾁｬ ﾉ ﾏｰﾁ"
1390 GOSUB 700
1400 CFLASH 0
1410 RETURN
1420 CLS
1430 CFLASH 1:COLOR 3
1440 LOCATE 8,8:PRINT"N O W  P L A Y I N G"
1450 LOCATE11,10:PRINT"MARCH-2"
1460 LOCATE12,12:PRINT"ｵﾓﾁｬ ﾉ ﾏｰﾁ  ﾘｽﾞﾑ ﾂｷ "
1470 GOSUB 930
1480 CFLASH 0
1490 RETURN
```

MUSIC の部分に，あまり本格的なプログラムをのせてなかったので，何曲か作ってもらいました。X1の MUSIC 機能の素晴らしさを確かめてみてください。

それではまず曲目紹介。

1. **白鳥の湖**　　チャイコフスキー
2. **アルルの女**　　ビゼー
3. **おもちゃのマーチ**Ⅰ
4. **おもちゃのマーチ**Ⅱ

おもちゃのマーチⅡでは，Ⅰにドラムの伴奏をつけてみました。SOUND 命令を使って，SHOT GUN の音とか作ってみたでしょ。あのやり方で，ドラムの音もできちゃったんですって！

```
SOUND 7, & B 0 0 0 1 1 1 0 0
PLAY "A : B : C"
```

3和音まで重ねられるわけですけど，ひとつを NOISE（ドラムの音）にしたので，メロディの部分は2和音です。

ノイズの説明は，もうちょっと必要かナ？

```
            ①      ②
          ┌───┐ ┌───┐
& B 0 0 0 1 1 1 0 0
    └─┘   | | | | | |
          C B A C B A
& B 0 0 0 0 0 1 1 1   ノイズ
& B 0 0 1 1 1 0 0 0   音
```

＆B 以降の表示で，NOISE にするか音にするかが決まります。上の図にパターンが3種類書いてありますよね。一番上から説明していきましょう。

これが実際のおもちゃのマーチのパターンで，①と②の部分が問題になるワケです。

①の011の0で，まずCの NOISE スイッチが ON になり，②の1で実行されます。つまり，最初の ┌──┐ の中の0は NOISE 準備，1は音準備，2番目の ┌──┐ の中の0は音実行，1は NOISE 実行なんです。

A：B：C 3つとも NOISE（つまり NOISE 3和音）にしたのが次のパターンで，3つとも音にしたのが3番目のパターンです。

NOISE の入れ方ひとつで，曲の感じもガラッとかわりますから，いろいろ調節してみてくださいネ！

それから，ボリュームのことには全く触れてませんでしたが，ボリュームは0から15までの数字で指定します。数字の大きい方がボリュームも大きくなり，0だと何も聞こえません。

PLAY "V 15：V 12：V 13"

こんな風に指定してやればOK。第1パートはボリューム15，第2パートは12，第3パートは13と少しずつボリュームに差をつけて演奏させる事も可能なんです。

まあ，とにかく聞いてみてください！

（P120より）

男性の精神機能は大脳の右半球あるいは左半球のどちらかによって，別々にコントロールされているのに対し，女性の精神機能は，両方の脳半球にまたがっているらしいことが知られています。

女性は一般的に，男性よりも成熟するのが早く，二つの脳半球が機能的に分化する時間が十分に与えられていないのではないか……ということのようです。

何か不思議な気持ち……でも安心してください。実際には男女間の平均的な違いよりも，同性の間の個人差の方がずっと大きいんですから……。

脳とホルモンには，関連性があると言われていますが，数年前，ロックフェラー大学のファーナンド・ノータボームとアーサー・アーノルドは，カナリアがさえずる機能をコントロールしている脳細胞の核の大きさが，雄と雌で違っているのを発見しました。雄の脳細胞の核は，雌のものより約四倍も大きかったそうですよ。

「きれいな声で鳴いているのが雄よ」と教わった覚えがあるけれど，こんなわけだったのねぇ。この核は発情期を境にして，大きくなったり，小さくなったりすることも確かめられています。

さらに，雌の成鳥にテストステロンという男性ホルモンのようなものを投与すると，さえずりを支配する脳細胞の核は，大きさが倍になり，雄と同じように鳴きはじめたんですって。

NOTE10
X1 HuBASICの概要
その最高のハードウエアと最高のソフトウエア

(1)パソコンテレビX１の持つハードウエアの諸機能

1)プログラマブル・ファンクションキー

X１のキーボード左上に，横長の KEY が５個見えますね。これがファンクション KEY です。このプログラマブル・ファンクション KEY には，HuBASIC スタート時に次のような命令が定義されています。

```
KEY LIST
KEY 1,"AUTO"+CHR$(13)
KEY 2,"?TIME$"+CHR$(13)
KEY 3,"KEY"
KEY 4,"LIST"+CHR$(26,13)
KEY 5,"RUN  "+CHR$(13)
KEY 6,"LOAD "+CHR$(13)
KEY 7,"WIDTH "
KEY 8,"CHR$("
KEY 9,"PALET "
KEY10,"CONT"+CHR$(13)
```

なお，これらの命令の意味と使い方は，本文中で説明しています。(P. 134参照)

2)専用カセットテープレコーダー

大切なプログラムを保存するための外部記憶装置として，オートマチックカセットテープが標準装備されています。このカセットは，プログラムですべての動作を行うことができ，AP-SS 自動頭出（機能）動作で，１本のテープの中に何のプログラムが記憶されているか，検索が容易に行えます。ただし，このカセットで音楽を聞くことはできません。

3)タイマー，カレンダー付クロック

年月日，時分秒を会話式に知らせてくれます。また，電源を切ってもこの記憶は消えません。

4)セントロニクス・インターフェース

世界的な標準仕様になってきたセントロニクスのインターフェースを内蔵しており，これによりプリンタ，プロッタなどすべての標準機器を接続することが可能です。

5)ＴＶコントロール

テレビのチャンネル，ボリュームなどをキーボードで切換えることができます。また，スーパーインポーズや，特殊効果などを楽しむことができます。

6)グラフィック・プライオリティ・ロジック

カラーグラフィックR，G，Bとキャラクターモードにおいて，おのおのに優先順位を付けることができます。キャラクター画面をグラフィック画面のうしろに隠したり，またその逆など，4種のスクリーン配置を自由にコントロールすることができます。

7)ユーザー定義，キャラクタゼネレータ

キャラクター，つまりキーボードから入力することのできる文字を，自分なりのオリジナル文字や記号に書きかえることが自由にできます。また，横2倍文字，縦2倍文字，縦横2倍文字なども簡単に表示することができます。

8)プログラマブル・サウンド・ゼネレータ

内蔵のスピーカから，和音も出力することができるようになり，簡単なミュージックシンセサイザーとして楽しむことができます。

(2)各種のグラフィック処理に対応する48Kバイトのグラフィック専用 RAM 実装

1)640×200 高分解能グラフィック

フルカラー8色　1画面または単色

(8色中1色選択) 3画面

2)320×200 マルチ画面

フルカラー8色　2画面または単色

(8色中1色選択) 6画面

(3)多彩な表示を実現した WINDOW 命令

WINDOW 命令によって論理座標を設定することにより，簡単なプログラムで，多彩な表示を実現できます。

(4)テキスト用 VRAM とグラフィック用 VRAM を別々に装備

テキスト用 VRAM とグラフィック用 VRAM を別々に持っているため，テキスト画面とグラフィック画面の合成を容易に行なえます。

(5)色を瞬時に変更できるパレット機能

パレットの概念によって，色の変更を瞬時に行なうことができます。

(6)簡単なプログラムで複雑な画面をコントロール

特殊なハードウエアと，これらをサポートする強力なソフトウエアにより，グラフィック画面同士や，またグラフィックとテキスト画面との間で，優先順位を持たせて重ね合わせることができます。この機能により，簡単なプログラムで複雑な動きの画面をコントロールすることができます。

(7)256 種のドットパターンを定義できるキャラクターゼネレータ RAM 実装

通常のキャラクター用 ROM の他に，ユーザーが自由に定義できるキャラクターゼネレータ RAM を持っており，255 種のドットパターンを定義することができます。

(8)レイアウトが自在にできる 4 種類のキャラクターサイズ

4 種類のキャラクターサイズをもっており，混在して用いることが可能です。組合せて使用することにより，見やすい画面を構成することができます。

(9)3 和音を合成できる充実したシンセ機能

プログラマブル・サウンドジェネレータ回路を持っており，各種の効果音を容易に作り出すことができます。

(10)プログラミングしながらテレビが観られるホームエンターテインメントの決定版

専用ＴＶに接続することにより，ＴＶ画面とコンピュータ画面を混在して扱うことができます。

⑾キーボードからテレビコントロールが可能

専用ＴＶのチャンネル，パワー，ボリュームなどをコントロールする命令をもっているため，これらのコントロールをプログラム中で行なえます。

⑿8つのタイマー機能内蔵

8つのタイマーを持っており，会話形式で，時刻設定が可能です。

⒀ジャンプ先にラベル指定を可能にし，プログラムを簡略化

プログラム中のジャンプ先として，ラベルを扱えるため，分かり易いプログラムを作ることができます。

⒁構造化プログラムに対応する各種コマンド

IF THEN ELSE/WHILE WEND/REPEAT UNTIL 等，構造化プログラムに対処した命令を備えています。

⒂専用カセットに対応するコントロール命令

専用カセットに対応した命令をいくつか用意しているので，プログラム中でも自由にカセットのコントロールができます。

⒃8桁まで有効な単精度実数

単精度実数の有効桁が標準で9ケタとなっているため，高精度の演算を少ない容量で実現できます。

⑴一般コマンド・ステートメント

AUTO ⇨ 行番号を自動的に発生させる

CLEAR ⇨ 変数の初期化及びメモリ領域の上限を設定する。メモリの上限アドレス

CONSOLE ⇨ テキスト画面のスクロール設定を行なう。上下方向スクロール開始行，上下方向スクロール行数，左右方向表示開始，左右方向表示の文字数

CONT ⇨ プログラムの実行を再開する

DELETE ⇨ プログラムの一部を削除する

EDIT ⇨ 指定行をリストする

LIST ⇨ プログラムを画面に表示する

LOAD ⇨ プログラムをメモリー内に取り込む

“ファイルディスクリプタ：ファイル名”

MARGE ⇨ 複数のプログラムを混合する

“ファイルディスクリプタ：ファイル名”

NEW ⇨ メモリ内のプログラムを抹消して変数もすべてクリアにする

RENUM ⇨ プログラムの行番号をつけ替える

ON/ERROR/GOTO ⇨ エラーの発生により分岐する

OPTION BASE ⇨ 配列の添字の下限を宣言する。0または1

RESUME ⇨ エラー回復後，プログラムの実行を再開する

ON/RESTORE ⇨ 指定されたいくつかの行番号に DATA 文を初期化する

ON/RETURN ⇨ 指定されたいくつかの行番号にリターンする

ON/RESUME ⇨ エラー回復後指定の行番号に復帰する

RUN ⇨ プログラムの実行を開始する

SAVE ⇨ メモリ内の BASIC プログラムを記録する

SEARCH ⇨ 指定文字列をプログラム中より探し出す

TRON ⇨ トレース機能を ON とする

TROFF ⇨ トレース機能を OFF とする

WIDTH ⇨ 画面に表示する文字数を変更する

1行の桁数は40または80

LOADM ⇨ 機械語ファイルをロードする

⇨ “ファイルディスクリプタ：ファイル名”，ロード番地

SAVEM ⇨ 機械語ファイルをセーブする

“ファイルディスクリプタ：ファイル名”，開始番地，終了番地

VERIFY ⇨ メモリ上のプログラムと外部デバイス上のファイルを比較する

“ファイルディスクリプタ：ファイル名”

LOAD? ⇨ VERIFY と同じ

LLIST ⇨ メモリ上にあるプログラムをプリンタに打出す

BOOT ⇨ IPL を起動する

CLR ⇨ 変数及び配列をすべてクリアする

LIMIT ⇨ BASIC で使用するメモリエリアを制限する

CALL ⇨ 機械語サブルーチンを呼び出す

DATA ⇨ READ 文で読み込むデータを用意する

DEF FN ⇨ ユーザーにより作られた関数を定義する

DEF INT ⇨ 変数の型を整数型とする

DEF SNG ⇨ 変数の型を単精度実数型とする

DEF DBL ⇨ 変数の型を倍精度実数型とする

DEF STR ⇨ 変数の型を文字型とする

DEF USR ⇨ 機械語ユーザー関数を定義する

DIM ⇨ 配列変数を定義し，メモリを割り当てる

END ⇨ プログラムの終了を宣言する

FOR ⇨ NEXT との間でループする

GOSUB ⇨ サブルーチンをコールする

GOTO ⇨ 指定した行へジャンプする

IF/GOTO/ELSE ⇨ 論理式の条件分岐を行なう

IF/THEN/ELSE ⇨ 論理式の条件判断を行なう

LABEL ⇨ プログラム中にラベルを付ける

LET ⇨ 変数に値を代入する（省略しても可）

NEXT ⇨ FOR ループの終端を示す

ON/GOTO ⇨ 指定されたいくつかの行へジャンプする

ON/GOSUB ⇨ 指定されたいくつかのサブルーチンへ分岐する

POKE ⇨ メモリ上の指定番地にデータを書き込む

REM ⇨ プログラムに注釈を入れる

REPEAT ⇨ UNTIL 文との間でループする

RESTORE ⇨ READ 文で読む DATA 文を指定する

RETURN ⇨ サブルーチンから復帰する

STOP ⇨ プログラムの実行を停止する

SWAP ⇨ 2つの変数の値を交換する

UNTIL ⇨ REPEAT 文の終端を示す

WHILE ⇨ WEND との間でループする

WEND ⇨ WHILE 文の終端を示す

(2)特殊命令

ERROR ⇨ エラーを発生させる（エラー番号0〜255）
ASK ⇨ 会話型のタイマー設定ルーチンを呼び出す
MID$ ⇨ 文字列の一部を置き替える
（MID$ の代入）
MEM$ ⇨ 指定メモリに対し文字列を直接転送する
MON ⇨ マシン語モニタに制御を移す
SCROLL ⇨ スムーズスクロールを ON/OFF する

(3)特殊関数と予約変数

PEEK ⇨ メモリ上の指定番地の内容を読み出す
FRE ⇨ メモリの未使用領域の大きさを与える
ERL ⇨ エラーの発生した行番号を与える
ERR ⇨ 発生したエラーコードを与える
USR ⇨ 機械語で作られたルーチンを呼び出す
VARPTR ⇨ 変数の格納されているメモリ番地を与える

(4)テキストに関する命令

CREV ⇨ テキストを反転モードにする
CFLASH ⇨ テキストを点滅モードにする
CSIZE ⇨ テキストの大きさを変える
CSRLIN ⇨ 現在のカーソルの行位置を与える

(5)テキストに関する関数と予約変数

POS ⇨ テキスト上のカーソルの水平位置を与える
MKI$ ⇨ 整数値を2文字の文字列に変換する
MKS$ ⇨ 単精度数値を5文字の文字列に変換する

MKD$ ⇨ 倍精度数値を８文字の文字列に変換する

ASC ⇨ 文字のキャラクタコードを与える

CHR$ ⇨ 指定したキャラクタコードを持つ文字を与える

HEX$ ⇨ 16進数に変換する

INSTR ⇨ 文字列の中から任意の文字列を探し，その位置を与える

LEFT$ ⇨ 文字列の左側から任意の長さの文字を与える

LEN ⇨ 文字列の総文字数を与える

MID$ ⇨ 文字列の中から任意の長さの文字列を与える

OCT$ ⇨ ８進数に変換する

RIGHT$ ⇨ 文字列の右側から任意の長さの文字列を与える

SPACE$ ⇨ 任意の長さの空白文字列を与える

STR$ ⇨ 数値を表わす文字列を与える

STRING$ ⇨ 任意の文字を任意の数だけ与える

VAL ⇨ 文字列の表わす数式を与える

MEM$ ⇨ メモリから文字を数個取り出す

HEXCHR$ ⇨ 16進数に変換する

BIN$ ⇨ ２進数に変換する

SPC ⇨ 任意の数だけ空白を出力する

TAB ⇨ 現在カーソルのある行の任意の位置まで空白を出力する

CHARACTER$ ⇨ 画面の座標（X, Y）に表示されている文字を与える

SCRN$ ⇨ 指定座標から指定数の文字をとってくる

INKEY$ ⇨ 現在押されているKEY データを与える

(6)グラフィック命令

SCREEN ⇨ グラフィック表示の入出力モードを指定する

GRAPH ⇨ SCREEN と同様

CLS ⇨ 画面をクリアする

COLOR ⇨ テキスト，バックの色指定をする

PSET ⇨ グラフィック画面の任意の位置にドットをセットする

PRESET ⇨ PSET と同様にリセットする

LINE ⇨ 指定された２点間に直線を引く

CIRCLE ⇨ 円を描く

PAINT ⇨ 指定された領域をぬりつぶす
POLY ⇨ 多角形を描く
WINDOW ⇨ グラフィック画面の表示エリアを指定する
POKE@ ⇨ VRAM 上の指定番地にデータを書き込む
GET@ ⇨ VRAM 上のデータを変数または配列に読み込む
PUT@ ⇨ VRAM 上にデータを書き込む
LAYER ⇨ テキストとグラフィックの優先順位を指定する
CANVAS ⇨ マルチ画面モード時の各グラフィック画面の色を指定する
PRW ⇨ テキストとグラフィックの優先順位を指定する
PALET ⇨ カラーパレットを変更する
POSITION ⇨ PATTERN 文によるパターン表示の位置指定をする
PATTERN ⇨ グラフィックエリア上に任意のグラフィックパターンを描く

(7)グラフィック関数

POINT ⇨ グラフィック座標の状態を与える
PEEK@ ⇨ VRAM 上の指定された番地の内容を読み出す
MIRROR$ ⇨ グラフィックパターンを指定された位置から反転する

(8)数値を取り扱う関数

ABS ⇨ 絶対値を与える
ATN ⇨ 逆正接を与える
CDBL ⇨ 倍精度数に変換する
CINT ⇨ 整数値に変換する
COS ⇨ 余弦を与える
CSNG ⇨ 単精度実数に変換する
EXP ⇨ e に対する指数関数の値を与える
FIX ⇨ 整数部を与える
INT ⇨ 小数点以下を切り捨てた数値を与える
LOG ⇨ 自然対数を与える
RND ⇨ 乱数を与える
SGN ⇨ 符号を調べる
SIN ⇨ 正弦値を与える

SQR ⇨ 平方根を与える

TAN ⇨ 正接値を与える

PAI ⇨ πのX倍の大きさを与える

RAD ⇨ 引数をX度としたときのラジアンを求める

CVI ⇨ 2文字の文字列を数値データに変換した値を与える

CVS ⇨ 5文字の文字列を数値データに変換した値を与える

CVD ⇨ 8文字の文字列を数値データに変換した値を与える

(9)プログラマブル・キャラクターゼネレータ

CGEN ⇨ キャラクターゼネレータからの読み出しを切換える

DEFCHR$ ⇨ ユーザー定義キャラクターゼネレータにパターンを定義する

(10)プログラマブル・キャラクターゼネレータの関数

CGPAT$ ⇨ ユーザー定義キャラクターゼネレータ

(11)KEYに関する命令

KEY0 ⇨ 先行入力バッファを定義する

KEY ⇨ ファンクションキーに文字列を定義する

DEF KEY ⇨ KEYと同様

KEY LIST ⇨ ファンクションキーの定義状態を表示する

KLIST ⇨ KEY LISTと同様

KBUF ON ⇨ KEYの先行入力をONする

KBUF OFF ⇨ KEYの先行入力をOFFする

REPEAT ON ⇨ リピート機能をONする

REPEAT OFF ⇨ リピート機能をOFFする

KEY ON ⇨ ファンクションキーからの割込みを可能とする

KEY OFF ⇨ ファンクションキーからの割込みを不可とする

KEY STOP ⇨ ファンクションキーからの割込みを停止する

ON KEY GOSUB ⇨ ファンクションキーによる割込みの開始行を定義する

CLICK ON ⇨ キーのクリック音をONにする

CLICK OFF ⇨ キーのクリック音をOFFにする

(12)ファイル関係の命令

INPUT# ⇨ シーケンシャルファイルからデータを読み込む
LINE INPUT# ⇨ 1行をシーケンシャルファイルから読み込む
PRINT# ⇨ シーケンシャルファイルにデータを出力する
WRITE# ⇨ シーケンシャルファイルにデータを書き出す
OPEN ⇨ ファイルを開く
CLOSE ⇨ ファイルを閉じる
DEVICE ⇨ ファイルディスクリプタのディフォルト値を決める

(13)ファイル関係の関数

EOF ⇨ ファイル終了コードを与える
LOC ⇨ ファイル中での論理的な現在位置を与える
FPOS ⇨ ファイル中での物理的な現在位置を与える
INPUT$ ⇨ 指定されたファイルより指定された長さの文字を与える

(14)入出力関係

INPUT ⇨ 変数を入力する
LINE INPUT ⇨ 1行を文字変数に入力する
LOCATE ⇨ テキスト画面のカーソル位置を指定する
CURSOR ⇨ LOCATE と同様
OUT ⇨ 出力ポートに1バイトのデータを送る
PRINT ⇨ 画面に情報を出力する
PRINT USING ⇨ 指定された書式に従って出力する
READ ⇨ DATA 文で用意されたデータを読み込む
WRITE ⇨ 画面にデータを出力する

(15)入出力関係の関数

INP ⇨ 入力ポートから値を得る

(16)音に関する命令

BEEP ⇨ 内蔵ブザーを ON/OFF する

MUSIC ⇨ ストリングデータで指定された音を出す

TEMPO ⇨ MUSIC 文で演奏する音楽の演奏速度を指定する

SOUND ⇨ サウンド IC に直接データを書き込む

PLAY ⇨ 音階，テンポを指定する

(17)タイマー関係の命令

TIME$ ⇨ 内蔵クロックの時刻，タイマーを設定する

DATE ⇨ 内蔵カレンダの月日を設定する

DAY ⇨ 内蔵カレンダの曜日を設定する

(18)カセットに関する命令

FAST ⇨ カセットテープの早送りをする

REW ⇨ カセットテープの巻戻しをする

EJECT ⇨ カセット取出し口をオープンする

APSS ⇨ カセットテープファイルの先頭を検出する

CSTOP ⇨ カセットをストップする

CMT ⇨ カセットの状態を与える

(19)ディスクに関する命令

FILES ⇨ ディスク上に登録されているファイル情報を表示する

LFILES ⇨ ディスク上に登録されているファイル情報をプリンタに表示する

KILL ⇨ ディスクより指定ファイルを抹消する

(20)プリンタ命令

LPRINT ⇨ プリンタに情報を出力する

HCOPY ⇨ 画面のコピーをとる

LPOS ⇨ プリンタヘッドの現在位置を与える

(21)TV に関する命令

TVPW ⇨ TV のパワーを ON/OFF する

CHANNEL ⇨ TV のチャンネルを変更する

VOL ⇨ TVの音量を変更する

CRT ⇨ 画面の表示を変更する

(22)ジョイスティックに関する命令

STICK ⇨ ジョイスティックの状態を与える

STRIG ⇨ ジョイスティックのトリガーボタンの状態を与える

(23)エラーメッセージ

NEXT without FOR	1	NEXTがあるのに FORがありません
Syntax error	2	文法がまちがっています
RETURN without GOSUB	3	この RETURN は GOSUB で呼ばれておりません
Out of data	4	READ で読む DATA がありません
Illegal function call	5	範囲外のデータが入りましたよ
Overflow	6	数が大きすぎて計算できません
Out of memory	7	メモリの残りがありません
Undefined label	8	行番号もしくはラベル文が無くて行き先がわかりません
Subscript out of range	9	配列のカッコの中の値がおかしいですよ
Duplicate Definition	10	同じ変数を2回 DIM を取ってますよ
Division by zero	11	0で割ることはできません
Illegal direct	12	ダイレクト命令で使えません
Type mismatch	13	文字と数値をまちがえていますよ
String too long	15	文字が長すぎますよ
Too complex	16	計算式がむずかしすぎます
Can't continue	17	つづけて実行が出来ません
Undefined function	18	その FUNCTION は登録されておりません
No RESUME	19	RESUME を実行しないでプログラムが終わりました
RESUME without error	20	エラーが起きていないのに RESUME されましたよ

Illegal format	21	エラーメッセージの定義されていないエラーが起きましたよ
Missing operand	22	パラメータをつけ忘れていますよ
Line buffer overflow	23	1行に入る文字の限界ですよ
Bad screen mode	25	グラフィックメモリは，グラフィック命令以外に使えませんよ
UNTIL without REPEAT	26	REPEAT 文がないのに UNTIL は使えませんよ
Out of tape	27	カセットテープが入っておりません
Tape read ERROR	29	カセットテープからデータを正常に読むことができませんよ
Bad file mode	30	異ったモードのファイル参照はできませんよ
Out of stack	31	POP 命令を実行しようとしたが，スタックに何も入っていないじゃないですか
WHILE without WEND	32	WHILE ループに WEND がありませんよ
WEND without WHILE	33	WHILE がないのに WEND があるのはおかしいよ
Reserved feature	34	カセットもしくは，ディスクが用意されていなければ実行できませんよ
FOR without NEXT	35	FOR があるのに NEXT がありませんよ
Format over	36	PRINT USING で指定したフォーマットが長すぎて出力できませんよ
FIELD overflow	50	FIELD 文でランダムファイル内のレコード長が256以上になっていますよ
Device in use	51	外部装置は今使用中です
Bad file number	52	オープンされていないファイルや，起動時に指定されていないファイルは参照できません
File not found	53	LOAD, KILL または OPEN 命令で，ディスクに無いファイルは，参照できません
Already open	54	すでに OPEN されているファイルは，2度と OPEN できませんよ
Device I/O ERROR	56	入出力装置に異常発生です
File already exists	57	NAME で変更しようとしたファイル名は，すでに登録されていますよ
Device full	60	データ量が入出力装置の許容容量を越えてしまいましたよ
Input past end	61	end of file か，空ファイルを読んではいけませんよ
Bad allocation table	64	ディスクの中の FAT テーブルが壊れていますよ

Bad file descripter	65	ディスクリプタが違っていますよ
Bad record	66	レコード番号が違っております
File not open	71	ファイルは，Open しなければ使えませんよ
Write protected	72	カセットテープ，またはフロッピーディスクが録音できない状態になっております
Device offline	73	指定された入出力装置がつながっておりませんよ

これら48個のエラーメッセージが用意されており，プログラムのあやまりを的確に指示してくれます。

(24)コントロールコード(1)

CTRL +		出力コード HEX	CHR$(X)	内容	備考
@ 又は 、		00	0		
A	a	01	1	自動挿入モードにする	
B	b	02	2	カーソルを1語戻す	
C	c	03	3	実行を停止する	SHIFT + BREAK
D	d	04	4	初期状態にする	注1
E	e	05	5	カーソルから右を行の終りまで消す	
F	f	06	6	カーソルを1語進める	
G	g	07	7	ピッ!とベルをならす	
H	h	08	8	一文字抹消する	DEL
I	i	09	9	水平タブ	HTAB
J	j	0A	10	ラインフィードをする	
K	k	0B	11	カーソルをホームポジションへ移動する	HOME
L	l	0C	12	画面を消去する	CLR
M	m	0D	13	キャリッジリターンをする	⏎
N	n	0E	14	ライン挿入（アップスクロール）	
O	o	0F	15	ライン挿入（ダウンスクロール）	
P	p	10	16		
Q	q	11	17	一時停止を解除する	
R	r	12	18	一文字挿入する	INS
S	s	13	19	一時停止をする	BREAK
T	t	14	20	水平タブをセットする	
U	u	15	21		
V	v	16	22		
W	w	17	23	次の行と結合する	
X	x	18	24		

Y	y	19	25	水平タブを抹消する	
Z	z	1A	26	カーソル以下の画面をすべてクリア	
[	¦	1B	27		
¥	ǀ	1C	28	カーソルを右へ移動	→
]	¦	1D	29	カーソルを左へ移動	←
^	—	1E	30	カーソルを上へ移動	↑
—		1F	31	カーソルを下へ移動	↓

㊟ CTRL ＋ D を実行すると、次のすべての命令が瞬時に実行されます。〔PRINT CHR$(4)では行われません〕

```
CONSOLE 0,25,0,80 or 40
COLOR 7
SCREEN 0,0 or 1,1
CREV 0
CFLASH 0
LSIZE 0
CGEN 0
WINDOW (0,0)-(639 or 319,199)
PRW
PALET
SOUND 7,&H3F
SOUND 8,0
SOUND 9,0
SOUND 10,0
```

(25)コントロールコード(2)

CTRL ＋	内容	備考
フルキーボードの0	バックの色を 黒 にする	COLOR, 0
1	〃 赤 〃	COLOR, 1
2	〃 緑 〃	COLOR, 2
3	〃 黄 〃	COLOR, 3
4	〃 青 〃	COLOR, 4
5	〃 マゼンダ 〃	COLOR, 5
6	〃 シアン 〃	COLOR, 6
7	〃 白 〃	COLOR, 7
テンキーの 0	文字グラフィックの色を 黒 に指定する	COLOR 0
1	〃 赤 〃	COLOR 1
2	〃 緑 〃	COLOR 2
3	〃 黄 〃	COLOR 3
4	〃 青 〃	COLOR 4
5	〃 マゼンダ 〃	COLOR 5
6	〃 シアン 〃	COLOR 6
7	〃 白 〃	COLOR 7
／	キャラゼネのROM／RAMを切換える	CGEN
＊	点滅文字スイッチ	CFLASH
—	反転文字スイッチ	CREV

㊟フルキーボードの数字と、テンキーの数字はコードが異なっています。

(26)TV のダイレクトコントロール

SHIFT +	内容	備　考
テンキーの0	音声消去	
1	チャンネル1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
／	10	
*	11	
−	12	
＋	TV.コンピュータ混在モード(コントラストダウン)	CRT3
＝	TVのみ	CRT0
・	コンピュータのみ	CRT1
,	ボリュームノーマル	
↑	ボリュームアップ	
↓	ボリューム　ダウン	
→	チャンネルアップ	
←	チャンネルダウン	

㊟X1のTVモードはVHFでもUHFでも好きなチャンネルを好きなチャンネル番号にセットできます。

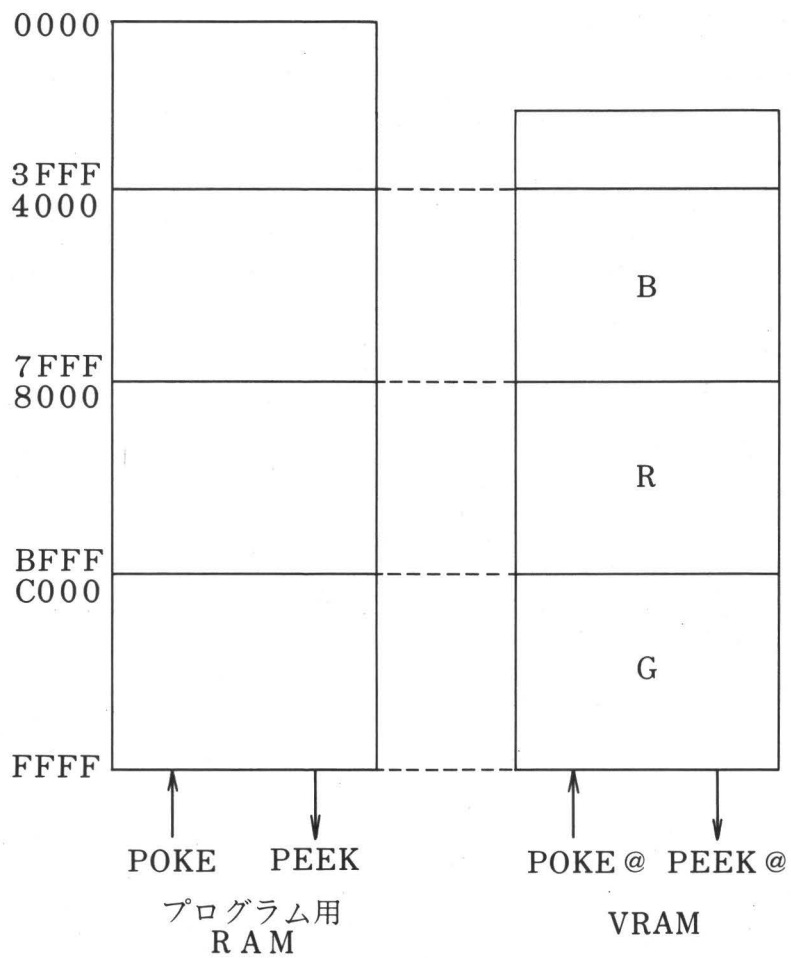
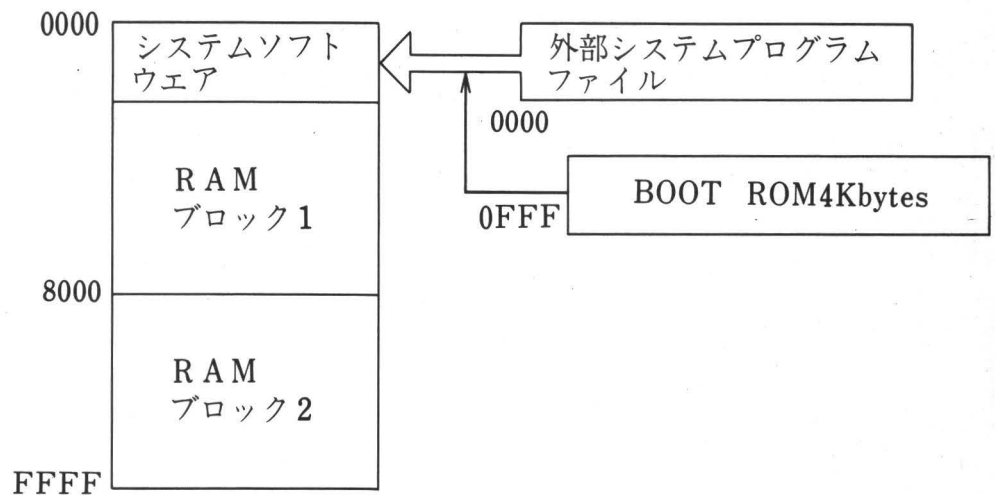
(27)ファイルディスクリプタ

ディスプレイ	CRT:
スクリーン	SCR:
キー	KEY:
プリンタ	LPT:
カセット	CAS:
ディスク0ドライブ	0:
ディスク1ドライブ	1:
ディスク2ドライブ	2:
ディスク3ドライブ	3:
グラフィックメモリ	MEM:
外部メモリ 0	EMM0:
～	
外部メモリ 9	EMM9:

すべての外部装置にこの様な名前を付けております。

(28)X 1 メモリマップ

X1メモリマップ
0000
3FFF
4000
7FFF
8000
BFFF
C000
FFFF
B
R
G
POKE
PEEK
POKE@
PEEK@
プログラム用
RAM
VRAM

IPL起動時のメモリマップ
0000
システムソフトウエア
外部システムプログラムファイル
0000
RAM
ブロック1
0FFF
BOOT ROM4Kbytes
8000
RAM
ブロック2
FFFF

(29)X 1 システム I/O マップ

I/O アクセスモードには，シングルアクセスモードと同時アクセスモードがあります。シングルアクセスモードは，いくつかの I/O のうちの個々にアクセスするモードです。同時アクセスモードは，同時に複数の I/O(グラフィック VRAM)に対してアクセスするモードです。

アドレス	シングルアクセスモード	同時アクセスモード
0000 0FFF	ユーザーI/Oポート	グラフィック VRAM (B・R・G)
1000 1FFF	システムI/Oポート	
2000 27FF	属性 VRAM	
3000 37FF	テキストVRAM	
4000 7FFF	グラフィック VRAM1(B)	グラフィック VRAM(R・G)
8000 BFFF	グラフィック VRAM2(R)	グラフィック VRAM (B・G)
C000 FFFF	グラフィック VRAM3(G)	グラフィック VRAM(B・R)

エピローグ

生まれて初めてコンピュータに触れて，勉強を始めてから3カ月が過ぎました。自分の全く知らなかった世界が，今はとても身近な感じ……。

でも，本当に不思議な気がするんですよね。小さなきっかけから始まったことが，3カ月の間，私の生活の中心になってしまったんですから。

そして，ここで私の得た教訓は，別にマイコンに限ったことではありませんが，少しでも興味とか関心を持ったのなら，まずやってみるのが一番。そして，やるのなら，やれる所までがんばり通す，ということ。何か始めたいんだけど，どうしようかしら？　なんて悩んでいる人がいたら，まず始めてみることです。

ところで，この本は本当に何も知らない私が書いた本なんですね。だから分かりにくい説明で困ってる人がいるんじゃないかって心配。それでも最後までつきあってくださった方……どうもありがとうございました。Bee先生も専門用語が使えず困ってましたけど，とにかく分かり易く教えてくださいました。私としては，コンピュータに何となく抵抗があるとか，いろいろなコンピュータの本を読んでもなおわからない……なんて人に，是非この本を読んでもらいたいなと思っています。

“入門書の入門書”だとでも考えてもらえばありがたいですね。

さて現在の私はというと，もうコンピュータはバッチリなんていう状態ではとてもなくて，相変わらず命令語を忘れたり，プログラムを作る時は，先生に教えてもらったりといったところ。

思うに，理解するということと，独力で考え出すということの間には，大きな隔たりがあるような気がします。私の場合は，やっと理解に手が届いたような所にいるわけで，すらすらプログラムが組めるようになるには，まだ当分勉強が必要でしょうね。

ただここで一つ強調しておきたいことは，今の私は早くそういう状態になりたいという気持ちが強くて，もうコンピュータへの不安感や，恐怖感はないと言うこと。

とっつきにくいのは，結局ほんの最初だけなんですね。だんだんおもしろくなってくることは受けあいます。だから，あなたにも是非始めて欲しいと思うんです。

HUDSON で働いているプログラマーの方々は，毎日頭を悩ませながらもとっても楽しそうです。早く本当の楽しさがわかるようになりたいなあ……。

ところで，私がこんな勉強を始めて周囲の反応はというと，友達はいいチャンスがあったネなんて言ってるし，両親は不思議な機械を買い込んで何を始めるつもり……なんて感じ。

友達の中には，文系でも情科研をとっている人もいるので興味を持って話しを聞いてくれます。（ウチの大学には情報科学研究所というものが設置されていて，そこでコンピュータを教えてもらえます。入門者はフォートランを勉強していますが，段階に応じて他の言語も勉強できるようです。毎週宿題が出されるので，みんな大変そう）あらためてコンピュータ人口の多いことに驚きますよねぇ。

さて，これから始めようという皆さんへ。

まず入門書とマニュアルを先生にして，必ず自分の手で KEY をたたきながら始めてみてください。コンピュータがなければ，本の中にも書きましたけど，秋葉原へ根気よく通うなんていう手もあるでしょう。

買いもしないのにお店の人に悪いかなぁ…なんて考えないで，どんどん聞いて覚えちゃうこと。今やマイコンショップの受け入れ体勢も変わってきています。お店の人を困らせちゃうぐらいになりましょう。顔なじみになってしまえば，もうこっちのもの!!

だいたい BASIC（他の言語でもいいんですけど）をマスターして，key 操作も分かるようになったら，次は人の作ったプログラムを研究してみましょう。ゲームなどは，あきるぐらい遊んで，ゲームのプログラムの内容を研究してみることです。

私が勉強していて特に実感したことは，とにかくプログラムを組むのは大変だということなんです。英語は読めても，なかなか英作文は書けないでしょ。コンピュータも同じです。むしろ作文以上に，プログラム作成は難しくて根気のいる作業です。

BASIC を知っている程度ではプログラムは組めません。ですから，しばらくは他人のプログラムを先生にして，なぜこうなっているのかな，と考えてみましょう。

次に，簡単なプログラムを書いてみることですね。たとえば数あてゲームなんか。誰かのプログラムに，少し手を加えて変えてみるぐらいでも，この段階ではいいんじゃないかなぁ。

そして，最後はとにかく悩み抜いて独力で作ってみること。何か参考にしながらやっていたのでは，結局自分のものになりません。最終的には自分で悩むしかないようです。

今までいろいろ書いてきましたけど，ここで一番大切な事を教えましょうか。

それは，コンピュータの勉強を好きになってしまうことです。毎日必ずコンピュータに触れること。触れなくては気がすまないという位になってしまうことですネ。せっかく勉強した事も，1カ月もすればだんだん忘れて来るものですよネ。だから，結局こういうことは，習慣にしちゃうのが一番なのです。

――と今の私が考えられる限りの方法を書いて来ましたが，別に勉強の仕方が決ってるわけじゃありませんヨ。

とにかく一番あなたに合った方法を見つけて下さい。ということなのね。

私にしても，まだほんの入り口に立っているようなもので，道はなお延々と続いています。もっともっと勉強して，また機会があれば，もう少し高度で，内容の濃いものを書いてみたいとも思っています。ご期待下さいね。

最後に，根気よくつき合ってくださった Bee 先生，そして Hudson の皆さん，本当にありがとうございました。また内容・文章表現ともに，稚拙なこの本につきあってくださった読者の皆さんにも心からお礼申し上げます。

1982年11月1日　品川ゆり

好評発売中
サンプル・プログラムがそのまま、カセットになりました。
SHARP パソコンテレビ X1
HuBASIC
私の勉強ノート
アプリケーション・プログラム
●日本国憲法
●世界の国々
●テレフォン・リスト
●ミュージック・プログラム
HUDSON GROUP HUDSON SOFT
品川ゆり・Dr.Bee COPYRIGHT© 1982
株式会社ラジオ技術社
〒101 東京都千代田区神田淡路町1-9
TEL03-251-0498
●このソフトはグラフィックRAMが必要です
SERIAL No.RG-1001-G ¥3,000
SHARP パソコンテレビ X1
アプリケーション・プログラム
¥3,000
株式会社ラジオ技術社
〒101 東京都千代田区神田淡路町2-1 TEL03-251-0498

1982年12月1日初版　1985年9月1日8刷

品川ゆり
Dr. Bee

発行人――金井　稔
株式会社ラジオ技術社
〒101 東京都千代田区神田淡路町2-1 TEL03-251-0498

定価　1,500円

2054-005030-8807